EXTRAÇÃO NÃO SUPERVISONADA DE DADOS DA WEB UTILIZANDO ABORDAGEM INDEPENDENTE DE FORMATO

ANDRÉ LUIZ LOPES PORTO

EXTRAÇÃO NÃO SUPERVISONADA DE DADOS DA WEB UTILIZANDO ABORDAGEM INDEPENDENTE DE FORMATO

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal do Amazonas como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Altigran Soares da Silva

Manaus

Novembro de 2015

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Porto, André Luiz Lopes
P853e Extração não supervisionada de dados da web utilizando abordagem independente de formato / André Luiz Lopes Porto. 2015
55 f.: il.; 31 cm.

Orientador: Altigran Soares da Silva
Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas.

1. Extração de dados. 2. Páginas Web. 3. Comércio Eletrônico. 4. Descrições de Produtos. 5. Alinhamento de dados. I. Silva, Altigran Soares da II. Universidade Federal do Amazonas III. Título

Agradecimentos

Aos meus pais, meu irmão e a minha namorada que com muito carinho e apoio, não mediram esforços e acreditaram em mim para que eu chegasse até esta etapa de minha vida.

Ao professor Altigran pela orientação, incentivo e oportunidades que tornaram possível a conclusão desta dissertação, e também aos meus outros professores de graduação e pós-graduação pelo conhecimento transmitido.

Aos meus amigos de universidade que me auxiliaram direta ou indiretamente nos meus estudos e na execução dos trabalhos acadêmicos.

Aos meus amigos de trabalho que sempre colaboraram com a troca de conhecimento e de experiências profissionais.

"Se as coisas não saíram como planejei posso ficar feliz por ter hoje para recomeçar. O dia está na minha frente esperando para ser o que eu quiser. "

(Charles Chaplin)

Resumo

Nessa dissertação de mestrado propomos um novo método para extração em páginas Web ricas em dados que utiliza apenas o conteúdo textual destas páginas. Nosso método, chamado de FIEX (**F**ormat **I**ndependent Web Data **Ex**traction), é baseado em técnicas de extração de informação por segmentação de texto, e consegue extrair dados de páginas Web nas quais métodos do estado-da-arte baseados em técnicas de alinhamento de dados não conseguem devido à inconsistência entre a estrutura lógica das páginas Web e a estrutura conceitual dos dados nelas representadas. O FIEX, diferentemente dos métodos previamente propostos na literatura, é capaz de extrair dados apenas utilizando o conteúdo textual de uma página Web em cenários desafiadores como casos severos de elementos textuais compostos, nos quais diversos valores de interesse para extração estão representados por apenas um elemento HTML. Para realizar a extração dos dados de páginas Web, o FIEX, é baseado em técnicas de eliminação de ruídos por redundância de informação e um método de extração de informação por segmentação de texto conhecido na literatura como ONDUX *(On-Demand Unsupervised Learning for Information Extraction)*. Em nossos experimentos, utilizamos várias coleções de páginas Web de diferentes domínios de produtos e de lojas de comércio eletrônico com objetivo de extrair dados de descrições de produtos. A escolha desse tipo de página Web, deve-se à grande quantidade de dados destas páginas estarem contidos em casos severos de elementos textuais compostos. De acordo com os resultados obtidos em nossos experimentos em diferentes domínios de produtos e lojas de comércio eletrônico, validamos a hipótese de que a extração baseada em apenas características textuais é possível e eficaz.

Palavras-chave: Extração de dados, Páginas Web, Comércio Eletrônico, Descrições de Produtos, Alinhamento de dados.

Abstract

In this thesis we propose a new method for extraction data in rich Web pages that uses only the textual content of these pages. Our method, called FIEX (**F**ormat **I**ndependent Web Data **Ex**traction), is based on information extraction techniques for text segmentation, and can extract data from Web pages where methods of state of the art based on data alignment techniques fail due to inconsistency between the logical structure of Web pages and the conceptual structure of the data represented in them. The FIEX, unlike the methods previously proposed in the literature, is able to extract data using only the textual content of a Web pages in challenging scenarios such as severe cases of textual elements compounds, in which various values of interest for extraction are represented by only one HTML element. To perform the extraction data of the web pages, FIEX is based on techniques of elimination noise by information redundancy and an information extraction method for text segmentation known in the literature as ONDUX *(On-Demand Unsupervised Learning for Information Extraction)*. In our experiments, we used various Web pages collections of different areas of products and e-commerce stores with goal to extract data from product descriptions. The choose of this type of Web page, due to the large amount of data these pages are contained in severe cases of textual elements compounds. According to the results obtained in our experiments in various areas of products and e-commerce stores, we validate the hypothesis that the extraction based on only textual features is possible and effective.

Keywords: Data Extraction, Web pages, E-commerce, Product Descriptions, Data Alignment .

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

IETS – *Information Extraction by Text Segmentation*
SGBD – *Sistema de Gerenciamento de Banco de Dados*
HTML – *HyperText Markup Language*
DOM – *Document Object Model*
STM – *Simple Tree Matching*
VINTS – *Visual Information and Tag Structure*
HMM – *Hidden Markov Model*
CRF – *Conditional Random Field*
ONDUX – *On-demand Unsupervised Learning for Information Extraction*
PSM – *Positional and Sequence Model*
FIEX – *Format Independent WebData Extraction*
CSS – *Cascading Style Sheets*
HIWE – *Hidden Web Exposer*
DELA – *Data extractionand label assignment*
JUDIE – *Joint Unsupervised Structure Discovery and Information Extraction*
UFAM – *Universidade Federal do Amazonas*
URL – *Uniform Resource Locator*
XML – *Extensible Markup Language*
Rel. – *Relevantes*
Ext. – *Extraídos*
Inc. – *Incorretos*
Cor. – *Corretos*
P. – *Precisão*
R. – *Revocação*
MF. – *Medida F*

Sumário

Capítulo 1

Introdução

A Web é abundante em páginas ricas em dados de interesse para vários tipos de usuários. Este é o caso de páginas de comércio eletrônico, bibliotecas digitais, catálogos de endereços postais e outros. Apesar de iniciativas recentes propostas por grandes companhias no sentido de representar dados estruturados de forma explícita em páginas Web [Bizer et al., 2013, Blog, 2011], o fato é que a grande maioria destas fontes em geral disponibilizam seus dados nestas páginas de forma apenas implícita, utilizando linguagem HTML ou outros formatos equivalentes, que são voltadas primariamente para consumo humano. Isso torna difícil e sujeita a erros a utilização em larga escala dos dados contidos nestas páginas.

Desta forma, existe a necessidade de se desenvolver métodos para extrair dados de páginas Web e estruturá-los automaticamente de forma correta, escalável e sem a intervenção humana. Para isso, inúmeros trabalhos apresentados na literatura nas últimas décadas propõem métodos de extração através da identificação de padrões de estruturação dos dados de interesse em páginas Web fornecidas como entrada. A lista de trabalhos publicados recentemente sobre o tema é bastante extensa. Assim, indicamos ao leitor interessado três artigos que trazem um mapeamento dos principais trabalhos nesta área: [Laender et al., 2002, Chang et al., 2006, Ferrara et al., 2014].

O estado da arte em métodos para extração de dados de páginas Web é representado por métodos que utilizam algoritmos de alinhamento de dados [Lu et al., 2007a, Zhai and Liu, 2005, Liu et al., 2010, Wang and Lochovsky, 2003, Lu et al., 2013]. Estes algoritmos tentam estabelecer correspondências ou alinhamentos entre nós de subárvores da árvore DOM de duas ou mais páginas fornecidas como entrada. O alinhamento é feito tomando como base características estruturais comuns entre os nós, ou seja, características derivadas de sua posição relativa na árvore DOM. Estes métodos assumem que os nós que sejam alinhados tem a mesma semântica e portanto denotam

a presença de itens de dados comuns entre elas. A principal característica que motiva o interesse por estes métodos é o fato de eles serem não-supervisionados, pois eles recebem como entrada somente as páginas Web de onde os dados serão extraídos, sem nenhum tipo de marcação ou anotação feita manualmente por usuários.

De forma geral, estes métodos são eficazes quando se assume que cada unidade de dado de interesse para extração encontra-se delimitado por elementos da estrutura HTML da página que os contêm. Embora implícitos, se estes delimitadores são usados de forma consistente na geração da página, eles podem ser identificados através da mineração de padrões comuns detectados nas páginas Web de entrada.

Em [Lu et al., 2007b], os elementos de estrutura HTML que contém uma única unidade de dados a ser extraído são chamados de *elementos textuais atômicos* (*atomic text nodes*). A Figura 1.1 apresenta um exemplo de um página que usa elementos textuais atômicos. Note que cada unidade de dado de um endereço postal é delimitado por um tag HTML.

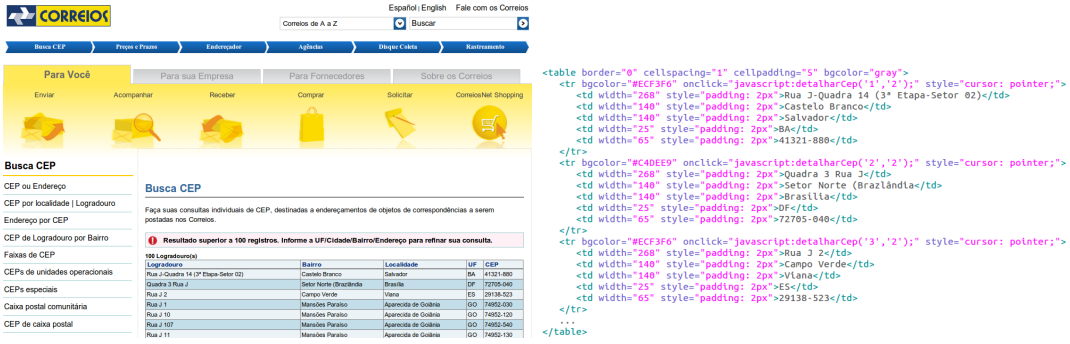

Figura 1.1. Exemplo de página e estrutura HTML com elementos textuais atômicos. Cada valor de atributo de um endereço postal é delimitado por um elemento HTML.

Por outro lado, existem casos em que um mesmo elemento HTML contém vários itens de dados. Neste caso, temos os chamados *elementos textuais compostos* (*composite text nodes*) [Lu et al., 2007b]. A maioria dos métodos existentes na literatura não consegue separar adequadamente os itens de dados individuais neste caso, pois não há separadores implícitos que sejam usados de forma regular. Um exemplo de uma página Web rica em dados que usa elementos textuais composto é apresentado na Figura 1.2. A página apresenta dados de ofertas de emprego. Nota-se que dois itens de dados: a descrição da vaga para emprego e o local da vaga, estão representados dentro um mesmo elemento HTML.

Nesta dissertação de mestrado, nosso objetivo é lidar com casos severos de elementos textuais compostos, onde, na verdade, todos os itens de dados em um mesmo

Figura 1.2. Exemplo de uma página HTML que usa elementos textuais compostos. Os itens de dados sobre a descrição da vaga para emprego e o local da vaga, estão representados dentro um mesmo elemento HTML.

objeto estão representados em único elemento de estrutura HTML. Esta situação é extremamente comum em páginas de sites de comércio eletrônico que disponibilizam ofertas de produtos. Um exemplo de casos como este é apresentado na Figura 1.3.

Figura 1.3. Página de Ofertas de Produto e estrutura HTML:casos severos de elementos textuais compostos

Para estes casos severos de elementos textuais compostos, os métodos atuais propostos na literatura para extração de dados em paginas Web encontrarão dificuldades, mesmo aqueles que são voltados para tratamento de elementos textuais compostos, tais como [Lu et al., 2007b, Pereira and Silva, 2006, Pedralho and da Silva, 2001, Zhai and Liu, 2005, Liu et al., 2010].

Isso ocorre porque existe uma inconsistência entra estrutura lógica das páginas Web e a estrutura conceitual dos dados nelas representadas. A estrutura lógica é representada pela árvore DOM das páginas Web, enquanto que a estrutura conceitual é representada pelos valores de atributos dos dados de interesse para extração, ou seja, os dados de interesse estão representado em um único nó da árvore DOM. Nestes casos, os métodos baseados em alinhamento não conseguem extrair os dados.

Por outro lado, existem na literatura diversos métodos para tratar situações similares a esta encontrada em elementos textuais compostos onde não é possível identificar delimitadores explícitos ou implícitos para os itens de dados que se deseja extrair. Este é o caso dos métodos de Extração de Informação por Segmentação de Texto (IETS) que se baseiam em modelos probabilísticos que tentam estimar a probabilidade de um segmento de texto representar um valor de um atributo [Agichtein and Ganti, 2004, Cohen and Sarawagi, 2004, Sarawagi, 2008, Cortez et al., 2010a, Cortez et al., 2011]. Embora eficientes, estes métodos assumem que sua entrada será fornecida em termos de trechos de texto, cada trecho representando um registro implícito de onde os dados de interesse serão extraídos por segmentação. Um exemplo deste tipo de entrada é apresentado na Figura 1.4 (esq.). No caso especifico do método JUDIE [Cortez et al., 2011], os registros não precisam estar explicitamente separados, como ilustrado na Figura 1.4 (dir).

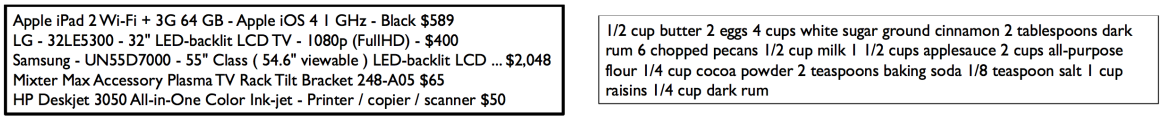

Figura 1.4. Exemplos de entradas aceitas pelos sistemas típicos de Extração de Informação por Segmentação de Textos.

Assim, embora estes métodos possam ser aplicados aos casos severos de elementos textuais compostos tais como os ilustrados na Figura 1.3, seria necessário antes processar as páginas Web, isolar os elementos textuais de interesse e somente então executar a extração dos dados por segmentação.

O método proposto nesta dissertação se enquadra na família de métodos não supervisionados de extração de informação por segmentação por texto. No entanto, ele pode receber como entrada uma página Web rica em dados como as ilustradas nas Figuras 1.1, 1.2 e 1.3. Para isso, utilizamos técnicas de eliminação de ruído por redundância de informação e técnicas de extração de informação por segmentação de texto (IETS).

Método Proposto

A inconsistência entre a estrutura lógica das páginas e a estrutura conceitual dos dados de interesse, é muito comum em páginas Web ricas em dados. Por isso, nesta dissertação, generalizamos os métodos que utilizam alinhamento dados como base para extração. Para isso, utilizamos técnicas de extração de informação por segmentação de

texto [Borkar et al., 2001, Lafferty et al., 2001a, Cortez et al., 2010a](IETS). Desenvolvemos um método chamado de FIEX (**F**ormat **I**ndependent WebData **Ex**traction), que baseado em técnicas de IETS, consegue extrair dados de páginas Web nas quais métodos baseados em técnicas de alinhamento de dados não conseguem.

O FIEX utiliza como base da extração o método ONDUX [Cortez et al., 2010a], que é um método probabilístico não supervisionado que extrai informação utilizando técnicas de segmentação de texto. Este método tem como entrada um conjunto de registros que serão segmentados e para cada segmento, um atributo será designado. No FIEX, de forma diferente do ONDUX, o método recebe como entrada páginas Web. A utilização de técnicas de IETS nos fornece a possibilidade de utilizar apenas o conteúdo textual de páginas Web, eliminando qualquer dependência da estrutura HTML das páginas Web.

Sendo assim, o FIEX necessita somente das páginas Web de entrada para extrair os dados. Para esta extração de dados, são executados 3 passos principais: (1) remover todas marcações HTML das páginas Web, (2) a partir deste texto resultante da remoção das tags HTML, é necessário extrair apenas o conteúdo textual de interesse eliminando conteúdos irrelevantes à extração. A eliminação desses textos consiste em remover qualquer tipo de informação que não seja relevante para extração, isso inclui: links de navegação, elementos HTML, textos de menus, banners, imagens e etc. Essa etapa é realizada por um simples algoritmo de detecção de redundância [Gulhane et al., 2010]. No último passo (3), é realizada a extração de dados utilizando o método ONDUX [Cortez et al., 2010a].

Neste trabalho, realizamos experimentos como extração de registros e dados de páginas Web de comércio eletrônico. Os registros representam descrições de ofertas de produtos. Um exemplo deste tipo de página é ilustrado na Figura 1.5.

Os dados dos produtos são representados por atributos que compõem a descrição de uma oferta. A extração de dados de registros consiste em rotular termos das descrições de ofertas com valores de atributos. Essa extração pode ser visualizada na Figura 1.6.

Em nossos experimentos, apresentamos resultados do método FIEX aplicados em páginas Web de comércio eletrônico utilizando métricas conhecidas na literatura [Baeza-Yates and Ribeiro-Neto, 2011]. Os resultados dos experimentos revelam que nosso método é capaz de extrair registros e seus respectivos dados de páginas Web identificando corretamente seus atributos. Estes resultados são detalhados no Capítulo 4.

Em resumo, nossa principal contribuição consiste de um método capaz de extrair dados de páginas Web de forma robusta, que: (1) lida com casos severos de elementos

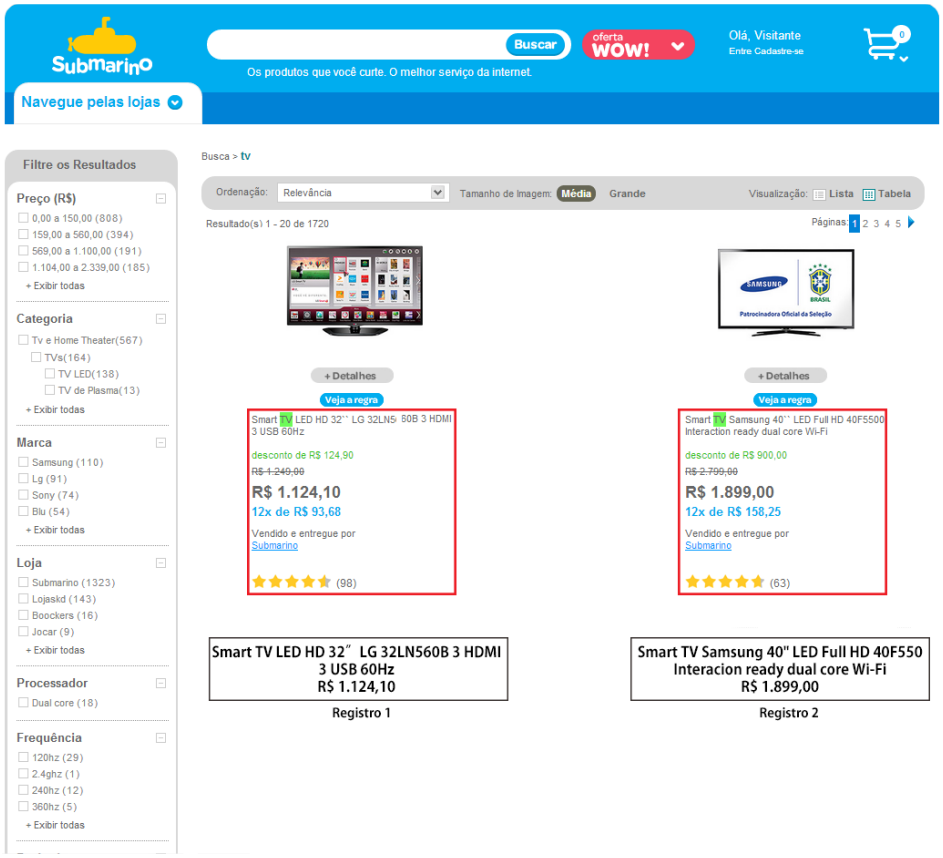

Figura 1.5. Exemplo de uma oferta de produto de um site de comércio eletrônico.

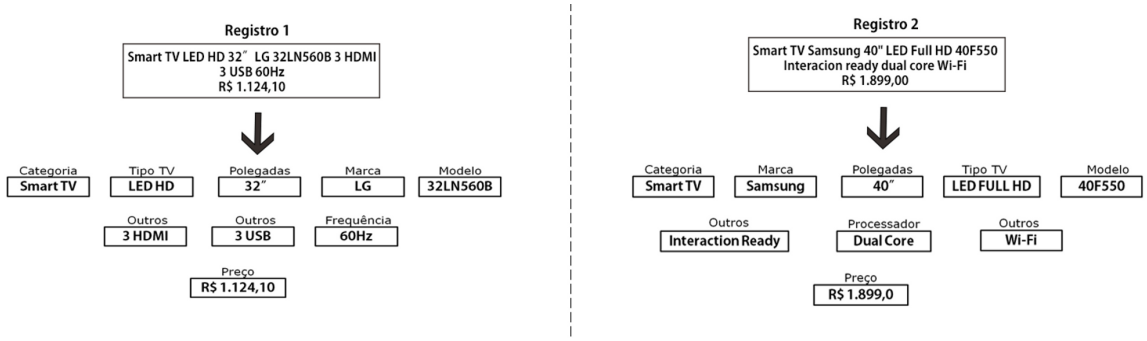

Figura 1.6. Exemplo de extração de dados em registros de produtos.

textuais compostos nos quais métodos que utilizam técnicas de alinhamento de dados são incapazes de realizar a extração devido a inconsistência da estrutura HTML e dos dados de interesse para extração. (2) não necessita de características de formato e estrutura das páginas Web para realizar a extração, utiliza apenas o conteúdo textual destas páginas; (3) os resultados dos experimentos apontam que o FIEX ao utilizar

apenas conteúdo textual, consegue extrair dados de páginas Web de forma eficaz; (4) não necessita de manutenção frequente e está menos suscetível a mudanças comuns de *layouts* destas páginas Web, diferentemente dos métodos de alinhamento de dados [Zhai and Liu, 2005, Raghavan and Garcia-Molina, 2001, Wang and Lochovsky, 2003, Lu et al., 2007a]; (5) a utilização de características textuais de nosso método auxilia na expansão da extração para outros domínios, tornando o FIEX um método flexível.

Organização da Dissertação

No Capítulo 2, apresentamos uma revisão dos principais trabalhos de extração de dados em páginas Web que utilizam técnicas de alinhamento de dados para a extração e os principais métodos da literatura sobre IETS. O Capítulo 3.6, é reservado para detalhar as três principais fases do funcionamento do método FIEX. Os experimentos realizados para verificar a eficácia do método FIEX e a metodologia utilizada nos experimentos são apresentados no Capítulo 4. Finalmente, no Capítulo 5 são apresentadas as conclusões e os possíveis trabalhos a serem desenvolvidos a partir do método FIEX.

Capítulo 2

Trabalhos Relacionados

Uma grande quantidade de métodos têm sido apresentados na literatura recente para extração de dados em páginas Web, por exemplo, [Selkow, 1977, Valiente, 2002, Reis et al., 2004, Constantine et al., 2005, Dalvi et al., 2009, Peters and Lecocq, 2013, Miao et al., 2009, Weninger et al., 2012]. Apesar disso, estamos interessados nos trabalhos que utilizam alinhamento de dados para fazer a extração, pois esses métodos possuem limitações e não funcionam para tipos de páginas que contenham casos severos de elementos textuais compostos. Nossa proposta é generalizar esses trabalhos utilizando técnicas de extração de informação por segmentação de texto (IETS), como base da extração. Nesse capítulo, estes trabalhos serão apresentados brevemente e comparados com nosso método.

2.1 Técnicas de alinhamento de dados

De forma geral, os métodos baseados em alinhamento de dados incluem duas fases distintas. Na primeira fase, são identificadas na página de entrada, registros implícitos que contém os dados de interesse. Esses registros correspondem à sub-árvores da árvore DOM da página de entrada, e nas suas folhas estão os dados de interesse. Na segunda fase, onde ocorre o alinhamento propriamente dito, são usados algoritmos que tentam estabelecer correspondências entre nós de duas ou mais destas sub-árvores procurando encontrar qual a semântica de cada um dos elementos textuais comuns a elas. Desta maneira, tenta-se distinguir "*strings*"que correspondem a valores de atributos referentes aos rótulos, delimitadores e etc.

O algoritmo avalia a similaridade entre duas árvores produzinho o maior casamento possível entre elas a partir de substuições e inserções nessas árvores. O algoritmo utiliza programação dinâmica para resolver esse problema.

Como um exemplo do processo de alinhamento de dados, apresentamos a seguir, o método proposto por [Zhai and Liu, 2005]. O processo é ilustrado na Figura 2.1. Na parte (a) desta figura, temos um conjunto S=$\{T_1, T_2, T_3\}$ de sub-árvores contendo registros para serem alinhados. Neste processo, a maior sub-árvore em número de nós do conjunto S, é escolhida como semente. No nosso exemplo, a sub-árvore T_1 foi eleita como semente, por isso $T_1 = T_S$

Na parte (b) desta figura, o primeiro alinhamento é iniciado entre T_S e T_2. Estas sub-árvores produziram apenas um casamento entre o nó b de cada sub-árvore, desta forma, não é possível saber aonde inserir os nós w, c, k e g, porque eles podem ser inseridos à direita ou à esquerda do nó d de T_S. Como a inserção desse nó é ambígua, a árvore é guardada no conjunto $R = \{\}$ para ser realinhada após uma nova inserção de nós em T_S. Continuando o alinhamento, na parte (c) da figura, T_S é comparada com a próxima sub-árvore T_3. Nesta comparação temos dois casamentos de nós b e d entre as sub-árvores, como ambos os nós existem em T_S, é possível determinar apenas um local para inserção desses nós. Então, os nós c, h e k podem ser inseridos em T_S de acordo com a posição deles em T_3 e T_S, ou seja, o nó c de T_3 será inserido entre b e d de T_S e os nós h e k de T_3 são inseridos à direita do nó d de T_S. Sempre que há uma inserção de nós, essa nova árvore T_S é comparada com as árvores guardadas do conjunto R que não obtiveram um casamento anterior, isso é realizado para verificar se ainda existe alguma possibilidade de alinhamento. Desta forma, comparando novamente T_S com o conjunto R (que contém apenas T_2), temos um casamento dos nós b e c. Com a nova árvore T_S resultante de T_S e T_3, agora é possível determinar aonde esses nós de T_2 serão inseridos em T_S. O nó w, foi inserido entre b e c, e os restantes k e g foram inseridos no final e à direita do nó c de T_S satisfazendo as antigas posições dos nós em T_S e T_2.

Como não há mais sub-árvores para prosseguir com o alinhamento, temos a árvore final resultante dos alinhamentos entre as árvores T_1, T_2 e T_3 representada por T_S na parte (d) da figura. Com essa árvore resultante do alinhamento, é possível organizar os nós de dados em uma tabela semelhante à tabela definida em um Sistema de Gerenciamento de Banco de Dados (SGBD) relacional, onde linhas representam instâncias e as colunas de dados representam valores de um mesmo atributo (alinhados). O preenchimento dos nós na tabela, são organizados de acordo com a posição de inserção, e à correspondência dele à árvore T_S. Na Tabela 2.1 é possível verificar a organização dos nós obtidos a partir do alinhamento final dos dados em nosso exemplo.

Existem também técnicas baseadas em alinhamento de dados que não usam árvores e que consistem em alinhar os dados por agrupamento [Liu et al., 2010, Pereira and Silva, 2006]. Na Figura 2.2, ilustramos um exemplo de alinhamento de dados utilizando essa abordagem.

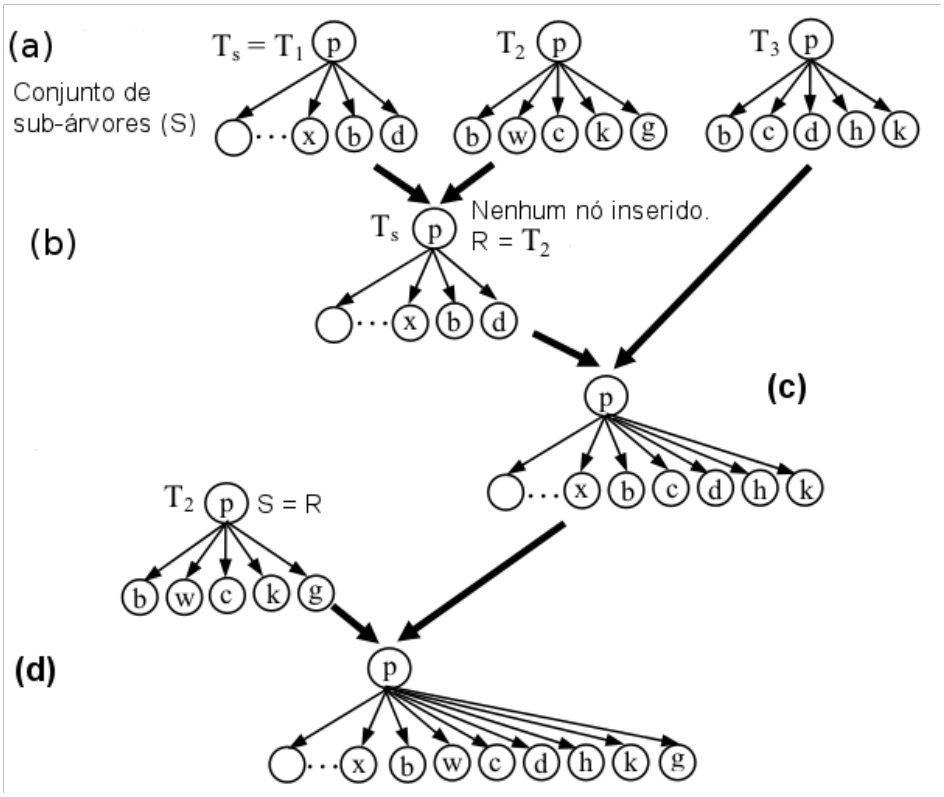

Figura 2.1. Alinhamento de árvores

	x	b	w	c	d	h	k	g
T_1	1	1			1			
T_2		1	1	1			1	1
T_3		1		1	1	1	1	

Tabela 2.1. Tabela final das árvores alinhadas ("1"significa uma unidade de dado)

Na parte (a) desta figura, existem 3 registros desalinhados $\{r_1, r_2, r_3\}$, cada linha é representada pelos registros e cada coluna representa um conjunto de unidades de dados que estão ou serão alinhados pela mesma semântica. Os símbolos geométricos são representações dos itens de dados. Símbolos com mesmo formato são resultado de algum algoritmo de agrupamento que adicionou tipos para os itens de dados de acordo com a semântica de cada um destes itens. Essa semântica, pode ser representada por uma combinação de características que envolvem principalmente características dos textos (cor da fonte, tamanho da fonte, tipo da fonte, posição no HTML e etc).

Inicialmente, na parte (a) da figura, temos as unidades de dados ($item_1^1$, $item_2^1$,$item_3^1$) com o mesmo símbolo, isso significa que estas unidades de itens apresentam mesma semântica e representarão uma mesma coluna na tabela final de alinhamento.

Na segunda coluna, ainda da parte (a), apenas as unidades de dados ($item_1^2$,$item_3^2$) são iguais. Para alinhá-los, os itens do registro r_2 são movidos para a direita e é adicionado um espaço em branco no $item_2^1$. Esse espaço em branco pode significar a ausência de algum atributo no registro. Com essa operação, na parte (b) da figura, esses itens foram divididos em dois novos grupos, C1={($item_2^1$, $item_2^3$)} e C2={($item_2^2$)}. Ainda na parte (b) da figura, a operação anterior de alinhamento realizada na coluna 2 de (a), por acaso, alinhou também a coluna 3 dos registros em (b). Continuando a interação do algoritmo, na coluna 4 da parte (b) da figura, existe um desalinhamento entre ($item_4^1$, $item_4^2$,$item_4^3$), pois o item ($item_4^1$) é diferente dos restantes itens. A mesma operação de alinhamento executado anteriormente é utilizada para realinhar esses itens. Ainda na parte (b), como o ($item_4^1$) de r_1 tem diferente semântica em relação ao itens da mesma coluna de r_2 e r_3, o algoritmo move os itens de r_1 para direita e adiciona um espaço em branco nele. Na parte (c) da figura, podemos ver o resultado desse alinhamento. O mesmo processo é executado nas colunas restantes dos registros e por fim, temos os dados alinhados na parte (d).

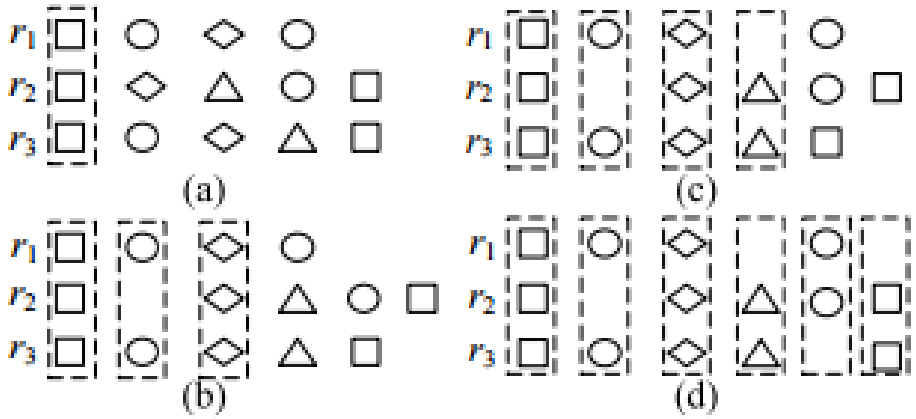

Figura 2.2. Alinhamento por agrupamento

Essa técnica, também pode ser utilizada para alinhar Árvores HTML e extrair os dados. Nesse caso, os registros de dados são considerados sub-árvores dá árvore DOM das páginas HTML, como mencionado anteriormente.

2.2 Métodos Baseados em Alinhamento de Dados

A seguir discutimos brevemente alguns dos principais trabalhos na literatura que se baseiam nessas técnicas de alinhamento de dados.

DELA [Wang and Lochovsky, 2003], é um método para extração automática de objetos de dados a partir de uma página Web que atribui rótulos aos dados. Esse tra-

balho, em particular, é concentrado em páginas Web que são resultados de consultas realizadas em formulários de pesquisa (*hidden web*). O sistema envia consultas para os formulários da página e automaticamente extrai objetos de dados das páginas do resultado das consultas. O método utiliza técnicas alinhamento de dados para posicionar os dados de forma alinhada em uma tabela e atribuir rótulos para as colunas dessa tabela.

O método é dividido em quatro principais fases: *form crawler*, *wrapper generator*, *data aligner* e *label assigner*. Na primeira fase (*form crawler*), é utilizado o método HiWe [Raghavan and Garcia-Molina, 2001] que coleta rótulos a partir dos elementos dos formulários presentes nas páginas Web e realiza consultas nesses formulários para obter resultados que contenham os objetos de interesses relacionados à respectiva consulta. O HiWe possui um banco de dados que guarda valores de algumas tarefas já realizadas e as utiliza como consultas. Esses valores utilizados nos formulários são usados também para auxiliar a fase de atribuição de rótulos (*label assigner*).

Na segunda fase, é utilizado um *wrapper* que irá induzir expressões regulares baseadas na estrutura das tags da página Web. Primeiramente o *wrapper* considera a página como uma sequência de *tokens* composta por tags HTML e um *token* especial chamado de "*text*"que representa qualquer texto dentro de pares de tags HTML. Depois, extrai as substrings repetidas dessas tag que estão presentes na sequência de *tokens* e induz uma expressão regular para a repetição de substrings de acordo com relações de hierarquia entre essas substrings.

Após a geração da expressão regular, na terceira fase (*data aligner*) acontece o alinhamento dos dados e a inserção deles em um tabela. Dado o *wrapper* induzido na fase anterior e as páginas Web, o alinhamento de dados, primeiramente extrai objetos de dados das páginas pelo casamento da expressão regular com a sequência de *tokens* de cada página. Os textos extraídos que casaram com a expressão regular, são colocados nas linhas de uma tabela. Após isso, os textos de cada coluna da tabela são separados, pois eles podem conter um ou mais valores que pertencem a outras colunas da tabela (atributos). Caracteres frequentes que não são letras nem dígitos de uma mesma coluna, são eleitos como separadores. A cada separação de valores, novas colunas nessa tabela são criadas. O problema dessa abordagem é que ela não trata atributos ausentes entre registros. Isso pode afetar drásticamente a extração final de dados devido ao desalinhamento deles.

A última fase (*label assigner*), utiliza heurísticas para atribuir rótulos para a tabela de dados gerada no alinhamento. A primeira, utiliza as consultas dos formulários para tentar encontrar casamentos entre a valores da consulta e valores da tabela. Se existir ocorrências desses valores, a coluna desses dados recebe o atributo do formu-

lário. A segunda, procura por rótulos presentes no cabeçalho das tabelas, no qual involuntarimente o cabeçalho é preenchido com os atributos de cada valor da tabela por desenvolvedores dessas páginas Web. A terceria, assume a premissa que alguns rótulos estão presentes juntamente com os valores dos atributos, desta forma, para cada coluna são procurados prefixos ou sufixos que se repetem nos dados. Se existir um prefixo repetitivo entre os dados da mesma coluna, ele será usado como atributo dessa coluna, e se for um sufixo, será utilizado para a próxima coluna caso ainda não tenha sido rotulada, pois existe grande probabilidade do sufixo pertencer ao atributo dos dados da coluna ao lado. A quarta, assume que alguns atributos possuem formatos convencionais (email, datas, preços e etc.). Se vários valores de uma mesma coluna da tabela apresentarem algum desses formatos, a coluna recebe esse rótulo. Essas heurísticas conseguem atribuir os rótulos para as colunas das tabelas finalizando o processo de extração.

Em [Zhai and Liu, 2005], para a extração dos registros, o método identifica Árvores de tags HTML que contém os dados de interesse. Para construir a árvore de extração, é utilizada uma estrutura aninhada das tags presentes no código HTML. No entanto, análises sofisticadas tem que ser incorporadas para lidar com erros no código HTML (por exemplo, falta de tags ou tags mal-formatadas). Neste trabalho, em vez de analisar o código HTML, a informação visual (os locais na tela em que tags aninhadas são apresentadas) é utilizada para inferir a relação estrutural entre tags e para construir uma árvore de dados. Este método conduz a construção de uma árvore mais robusta devido a alta tolerância a erros de renderização de navegadores da Web (por exemplo, Internet Explorer, Firefox, etc).

Para fazer a extração dos dados, uma técnica de alinhamento de dados é aplicada, a qual consiste em alinhar registros que são representado como sub-árvores da árvore DOM da página Web. Para isso, são encontradas árvores similares a partir de um algoritmo conhecido como STM (Simple Tree Matching) [Yang, 1991]. No final, uma tabela é preenchida com os dados extraídos onde as colunas representam atributos e as linhas representam os registros. Todo processo desse alinhamento foi explicado no início dessa seção e ilustrado na Figura 2.1, onde apresentamos um exemplo de alinhamento de três sub-árvores similares e a inserção desses dados em uma tabela.

Em [Lu et al., 2007a], uma técnica de alinhamento de dados é aplicada de forma holística dentro de registros implícitos que possuem a mesma semântica. Inicialmente, um processo de extração dos registros é realizado utilizando o algoritmo ViNTs [Zhao et al., 2005]. Desta forma, cada registro extraído é armazenado em uma estrutura de árvore com uma única raiz, onde cada nó da árvore corresponde a uma tag HTML ou a um pedaço de texto da página Web. Com essa estrutura, é possível

localizar cada nó na página HTML original e utilizar as relações entre os nós que contenham os dados. A posição física da informação de cada nó na página processada, incluindo suas coordenadas de área e tamanho, são obtidas utilizando o ViNTs.

Para realizar o alinhamento dos dados, são necessárias três fases. A primeira, chamada de *Align Text Nodes*, tem como objetivo criar grupos de dados a partir de um agrupamento de nós de texto com a mesma semântica. Para determinar a semântica dos dados, é utilizada uma combinação de cinco características baseadas no conteúdo dos dados (palavras-chaves repetidas), apresentação do texto (nome da fonte, tamanho da fonte, cor da fonte, fonte itálica e o peso da fonte), tipo de dados (data, hora, inteiro, decimal, porcentagem, moeda, e numéro ordinário), caminho dá árvore de tags (caminho dá raiz até a folha) e repetição de prefixos e sufixos.

A segunda fase do alinhamento de dados é chamada de *Split Composite Text Nodes*. *Composite Text Nodes* são nós que podem conter mais de um valor de diferentes atributos. Essa fase é responsável por separar os valores que representem um novo atributo. A etapa de separação é realizada do seguinte modo. Primeiro, um grupo de nós de texto não pode ser dividido quando uma das seguintes restrições é satisfeita: (a) um nó por completo possui um link; (b) os textos de todos os nós do grupo são os mesmos; (c) todos os nós no grupo têm o mesmo tipo de dado. Por exemplo, se todos os nós do grupo são do tipo de data, cada nó já é uma única unidade semântica (*atomic node*). Em seguida, o grupo é dividido se nenhuma das limitações acima referidas forem verdadeiras. Para fazer a divisão correta, o método identifica separadores de texto. Para essa identificação, observa-se que: (1) o mesmo separador é suscetível de ser usado para separar diferentes unidades semânticas dentro do mesmo nó de texto, e (2) o mesmo separador é susceptível de ser utilizado para separar os textos nas unidades de dados do mesmo conceito entre múltiplos registros. Com base nestas observações, é realizada uma primeira varredura nas seqüências de texto de cada nó de texto no grupo e selecionado um caracter (que não é uma letra e nem um dígito) com maior número de ocorrências, ele será eleito como separador. Para cada nó de texto dentro de um mesmo grupo, é feita a separação dos valores.

A fase final é chamada de *Align data units*, ela consiste em reagrupar os grupos da fase anterior e colocar as unidades de dados em seus respectivos registros. Desta forma, para cada grupo, um novo agrupamento é realizado com o objetivo de criar grupos de dados apenas com as similaridades semânticas do seu registro e não de todos, como anteriormente realizada na primeira fase. Para isso, o mesmo algoritmo da primeira fase é utilizado aqui, porém, existe uma nova informação de similaridade para combinar com as características apresentadas na primeira fase. O valor dessa característica é 0 se a unidade de dado pertencer a um registro diferente e é 1 se pertencer ao mesmo

registro. Com isso, o mesmo algoritmo de agrupamento da primeira fase, consegue agrupar as unidades de dados e transformá-las em registros.

Para finalizar a extração, é necessário atribuir rótulos aos dados extraídos. Nessa fase, são utilizados combinações de seis anotadores. Esse anotadores utilizam informações dos cabeçalhos extraídos de tabelas presentes nas páginas Web, consultas e atributos dos formulários, repetição de palavras-chaves, prefixos e sufixos frequentes e base manual de alguns atributos conhecidos. Para definir o rótulo final, o atributo que tiver maior probabilidade dessa combinação, será eleito como o rótulo daquele grupo de dados.

O trabalho apresentado em [Wang and Lochovsky, 2003] é o mais semelhante ao [Lu et al., 2007a]. Entretanto, a abordagem é significativamente diferente. O alinhamento do DELA consiste em um método que é puramente baseado em tags HTML, enquanto [Lu et al., 2007a] alinha os dados utilizando agrupamento com a combinação de várias características, tais como: tipo de dados, conteúdo de texto, e adjacência informações.

Em [Lu et al., 2013], temos uma extensão de [Lu et al., 2007a], no qual é realizada uma discussão significativamente mais abrangente sobre as relações entre os nós de texto e dados neles contidos. Especificamente, foram identificados quatro tipos de relacionamento: um para um, um para muitos, muitos para um e um para nenhum. O trabalho apresenta uma análise de cada tipo, sendo que apenas dois dos quatro tipos (ou seja, um para um e um para muitos) foram muito brevemente mencionados em [Lu et al., 2007a] e que também são utilizados em [Wang and Lochovsky, 2003]. Além disso, o algoritmo de alinhamento é significativamente melhor, pois uma nova etapa é adicionada para lidar com esses novos relacionamentos de nós e dados. Com estas duas melhorias, o novo algoritmo de alinhamento leva todos os quatro tipos de relacionamentos em consideração e apresenta resultados mais satisfatórios que o anterior. Ainda assim, o método depende da estrutura e características dos dados. Em nossa abordagem baseada apenas em informação textual, esses algoritmos e propriedades não são necessárias.

Em [Liu et al., 2010], é proposto um método chamado VIDE que cria uma árvore de blocos visuais utilizando algumas características do texto e das imagens. O algoritmo VINTS [Zhao et al., 2005] é utilizado para criar essa árvore visual de blocos. Cada bloco possui um tamanho de área que é computado a partir das coordenadas do tamanho do texto e das imagens. A página Web é composta por um bloco que contém outros blocos representados pelos registros.

Para extrair os registros, é utilizado uma técnica de alinhamento de dados utilizando a área de cada bloco. Na primeira fase, para eliminar os blocos que não são os

registros e encontrar a região dos dados de interesses, são utilizadas duas premissas: (1) as regiões de dados são sempre centralizadas horizontalmente; (2) o tamanho da região de dados é geralmente grande em relação ao tamanho de toda a área de página. A primeira premissa, pode ser aplicada diretamente, a segunda é representada por um *score* calculado a partir da área de um bloco b e a área total da página x, ou seja, (area$b$/ area$x$) $>$ *Tarea*. Aonde área b é a área do bloco b e área x é a área de toda página. O *Tarea* é um limiar escolhido a partir de séries de testes em páginas de exemplo da *deep web*. Se o *score* do bloco de comparação for maior do que o limiar ele será descartado.

Depois do descarte de blocos, é encontrado a região que contém os dados, porém, ainda é necessário eliminar blocos ruidosos que ainda aparecem no início e no fim dessa região. Para essa etapa, é utilizada uma premissa que consiste na posição do bloco, se o bloco não estiver alinhado à esquerda com nenhum dado, ele é removido. Esse alinhamento é realizado de acordo com a área de cada bloco. Blocos com tamanhos de áreas parecidas são alinhados. O problema dessa abordagem é que ela não garante total remoção desses blocos o que prejudica a extração final dos dados.

Após a eliminação de blocos ruidosos, os blocos presentes nessas região precisam ser agrupados para compor os registros. Nessa fase, é aplicado um algoritmo de agrupamento baseando na similaridade da aparência dos blocos. Essas aparências são características das imagens, textos e links. Para imagens, o foco, é o tamanho. Para o plano e links, o foco são as fontes compartilhadas entre eles. Se dois blocos são mais similares no tamanho da imagem e na fonte, eles devem ser aparentemente parecidos. O agrupamento é iniciado pelo primeiro bloco da página que é utilizado como semente para formar um grupo inicial. Para os blocos restantes é calculada a similaridade entre o primeiro grupo e o próximo bloco. Se a similaridade for maior que um limiar T, o bloco é adicionado a esse grupo. Se a similaridade for menor que um limiar T, é criado outro grupo. Esse limiar T é definido por um treino de páginas de exemplo. Com esse agrupamento, títulos, preços e outros valores de atributos são agrupados em grupos diferentes.

Esse agrupamento, ainda não garante a criação dos registros, os blocos do mesmo grupo são do mesmo tipo, mas de diferentes registros. Para isso, é necessário reagrupar os dados de forma que blocos do mesmo registro formem um grupo. Esse reagrupamento é realizado de acordo com a posição de dois blocos.

Na primeira fase, o algoritmo reorganiza os blocos em cada grupo, baseado na aparência e na ordem dos blocos na página Web. Na segunda fase, são criados n grupos inicializados com uma semente, sendo que n representa o tamanho do maior cluster da etapa anterior. Na terceira fase, é determinado qual bloco pertence a cada grupo.

Nessa fase, é verificada a posição e adjacências de cada bloco em relação aos outros blocos. Se dois blocos C(i,j) estão abaixo ou acima do outro, eles pertecem a grupos diferentes. Se C(i,j) estão na frente ou atrás um do outro, eles pertencem ao mesmo grupo. No final do agrupamento, temos todos os registros recontruídos.

Com a criação dos registros, é realizada o alinhamento de dados que consiste em preencher uma tabela na qual dados com a mesma semântica e de registros diferentes vão estar na mesma coluna. Para esse alinhamento, utiliza-se uma heurística baseada em características visuais. Dois itens de diferentes registros possuem a mesma semântica se a fonte ou a posição dos blocos são parecidas. Além disso, dois itens são casados se a posição absoluta dos blocos são próximas e possuem a mesma fonte (cor, tamanho e características). Por posição absoluta, compreende-se a distância entre o lado esquerdo da região de dados e o lado esquerdo da unidade de dados. Quando dois itens não possuem a mesma posição absoluta, eles ainda podem ser iguais se a posição relativa for igual. Isso é utilizado para casos em que existem atributos ausentes entre os registros.

Para fazer o alinhamento, considera-se inicialmente que todos os itens de dados de cadas registro não estão alinhados. Para cada bloco da unidade, de cada registro, é verificado se eles possuem alguma semelhança de acordo com a posição da unidade e a fonte do texto. Se possuir semelhança, os dados são alinhados. Se o próximo bloco, não possuir similaridades com os blocos vizinhos, é preenchido com espaco e branco e posicionado para a direita. Isso faz com que o bloco possa ser alinhado com outro bloco de outro registro. Isso é feito para todos os blocos. No final, temos todos os dados alinhados e unidades de dados faltosos, são preenchidos por espaços em branco. O método não adiciona rótulos para a tabela final, apenas a cria.

Apresentamos vários trabalhos que aplicam algoritmos em árvores DOM. Muitos necessitam que os dados estejam em uma árvore aninhada e sigam um padrão contendo símbolos ou caracteres para delimitar os dados. Nosso cenário de extração, possui todos os valores de atributos presentes em apenas uma tag HTML e ausência de delimitadores explícitos, o que limita a extração de registros e também a rotulação dos dados pelos trabalhos e algoritmos apresentados. Não é possível assumir tais padrões e características em nossos dados, pois, como discutido no Capítulo 1, em casos severos de elementos textuais compostos, não existem delimitadores para auxiliar a extração nem padrões na organização dos dados.

2.3 Métodos Baseados em Segmentação de Texto

Nessa seção, apresentamos brevemente os principais trabalhos que se baseiam em Extração de Informação por Segmentação de Texto (IETS), que é uma técnica amplamente aplicada para extrair valores de atributos que ocorrem em registros de dados semi-estruturados implícitos na forma de texto contínuo, como descrições de produtos, citações bibliográficas, endereço postal, anúncios classificados, etc. O objetivo principal é segmentar o texto de entrada e rotular uma sequência de dados. Os métodos atuais de IETS utilizam modelos probabilísticos baseados em grafos [Borkar et al., 2001, Lafferty et al., 2001a, Cortez et al., 2010a], em que os nós (estados) representam atributos e as arestas (transições) representam as estruturas dos registros de dados. Quando devidamente treinados, esses modelos são capazes de prever com precisão uma sequência de rótulos a ser atribuído à uma seqüência de segmentos de texto correspondente à valores de atributos. O processo de aprendizagem consiste na captura de características relacionadas com o conteúdo (estado), que caracterizam o domínio dos atributos (valores comuns, termos que as compõem, formato, apresentação, etc.), e características relacionados à estrutura, (posição e sequência dos valores de atributos, etc.), que caracterizam a estrutura dos registros dentro do texto de origem.

Hidden Markov Models (HMM) [Borkar et al., 2001] utiliza modelos que representam as distribuições de probabilidade sobre sequências de observações para o reconhecimento de valores de cada atributo. O HMM consiste em uma observação de um determinado espaço de tempo t definida pela variável Y_t, que podem ser símbolos de um alfabeto discreto, textos, variáveis reais, números inteiros, ou qualquer outro objeto, desde que possa ser definido uma distribuição de probabilidade sobre ele.

Por exemplo, suponha que as observações são amostradas em intervalos de tempo definidos, de modo que t possa ser um índice de tempo de valores inteiros. O modelo HMM recebe esse nome a partir de duas propriedades. Em primeiro lugar, ele assume que a observação no tempo t foi gerada por algum processo cujo estado S_t é escondido do observador (*hidden state*). Em segundo lugar, ele assume que o estado deste processo oculto satisfaz a propriedade de Markov (Markov Models), que consiste em, dado um valor de S_{t-1}, o estado atual S_t é independente de todo os estados, ou seja, o estado encapsula tudo o que precisa saber sobre a história do processo, a fim de prever o futuro dele. A saída também satisfaz uma propriedade Markov em relação aos estados, ou seja, dado dois estados, S_{t-1} e S_t, eles independem entre si e dos outros estados de acordo com observações em todos os outros índices de tempo.

Essa distribuição de probabilidade pode ser desenhada graficamente por uma rede *Bayesiana* ou modelo gráfico probabilístico que irá mostrar as depedências entre

as variáveis do modelo. Na representação gráfica, cada variável é representada por um nó no grafo, e cada nó recebe arestas diretas a partir de nós, no qual o nó é condicionalmente dependente. Na Figura 2.3, ilustramos o exemplo de um modelo HMM simples utilizando uma rede *Bayesiana*. O HMM é aplicado em uma extensa

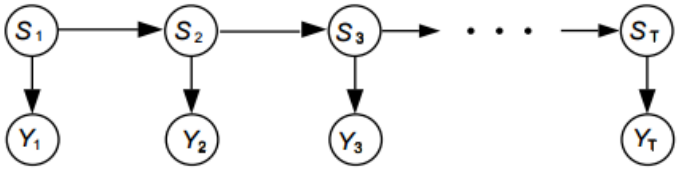

Figura 2.3. Exemplo de rede Bayesianda com relações independentes segundo HMM

quantidade de problemas: reconhecimento de voz, computação biológica, detecção de falhas, extração de informação e etc.

O CRF (Conditional Random Fields) [Lafferty et al., 2001b] é uma abordagem proposta como alternativa para o HMM para extrair valores de atributos de textos. Modelos CRFs são adequados para problemas de modelagem na qual as transições de estado e as emissões de probabilidades podem variar entre os estados ocultos, dependendo da sequências de entrada. Isso quer dizer que o CRF é, essencialmente, uma forma de combinar as vantagens de classificação e modelagem gráfica, combinando a capacidade de modelar dados variados de uma forma mais robusta, com a capacidade de utilizar um grande número de recursos de características de entrada para a previsão. O CRF pode ser também utilizado em várias aplicações: processamento de texto, computação biológica, visão computacional, etc.

O ONDUX [Cortez et al., 2010b] é um método probabilístico não supervisionado que extrai informação utilizando técnicas de segmentação de texto. A abordagem, depende de dados pré-existentes (*Knowledge Base*), compostos por um conjunto de atributos e valores de uma fonte de dados, para associar segmentos de *strings* da entrada com um determinado atributo dessa base. Na Figura 2.4 ilustramos um exemplo simples de uma base de conhecimento composta por um conjunto de atributos com seus respectivos valores.

O ONDUX possui três etapas principais. A primeira etapa do método é chamada de *Blocking*. Nesta etapa, os textos de entrada são segmentados em unidades, chamadas de blocos. Blocos são simples sequências de texto (palavras) que estão sujeitas a formar um ou mais valores de um determinado atributo. Esta etapa é baseada na co-ocorrência dos termos da entrada com valores de atributos da base de conhecimento.

$$KB = \{\langle Categoria., O_{Categoria}\rangle,\ \langle Marca, O_{Marca}\rangle, \langle Proc., O_{Proc.},\rangle,$$
$$\langle Mem.Ram, O_{Mem.Ram}\rangle, \langle Cor, O_{Cor}\rangle,\ \langle Tam.Tela, O_{Tam.Tela},\rangle\}$$
$$O_{Categoria} = \{\text{“}\mathbf{Notebook}\text{"},\text{“}\mathbf{Ultrabook}\text{"},\text{“}\mathbf{Laptop}\text{"}\}$$
$$O_{Marca} = \{\text{“}\mathbf{Dell}\text{"},\text{“}\mathbf{HP}\text{"},\text{“}\mathbf{Asus}\text{"}\}$$
$$O_{Processador} = \{\text{“}\mathbf{Intel\ Core\ I3}\text{"},\text{“}\mathbf{Intel\ Core\ I5}\text{"}\}$$
$$O_{MemriaRAM} = \{\text{“}\mathbf{4GB\ RAM}\text{"},\text{“}\mathbf{8GB\ Memória}\text{"}\}$$
$$O_{Cor} = \{\text{“}\mathbf{Preto}\text{"},\text{“}\mathbf{Prata}\text{"}\}$$
$$O_{TamanhoDaTela} = \{\text{“}\mathbf{15\ polegadas}\text{"},\text{“}\mathbf{10\ polegadas}\text{"}\}$$

Figura 2.4. Exemplo simples de base de conhecimento

Dado uma entrada de texto I contendo vários termos, cria-se blocos a partir da ocorrência de espaços em brancos nesses textos. Símbolos especiais e de pontuação são descartados. Para cada termo de I, é verificada a ocorrência do termo atual T_J e o próximo termo T_{J+1} na base de conhecimento, se ambos termos T_J e T_{J+1} ocorrerem nessa base, eles irão compor o mesmo bloco T_J. Caso contrário, um novo bloco será criado para T_J. Esse processo é repetido até que todos termos de I sejam assinalados a blocos. Apesar dessas unidades pertencerem a um mesmo valor de atributo, esses valores podem ter termos divididos entre dois ou mais blocos.

Na próxima etapa, chamada de *Matching*, os blocos são associados a valores de atributos previamente conhecidos que estão disponíveis nas bases de conhecimento. Nesta etapa, cada bloco recebe um rótulo com o nome do atributo para o melhor casamento encontrado entre os valores de texto do bloco e a base de conhecimento. Para isso, um conjunto de poucas funções de similaridade são utilizadas. Essas funções são escolhidas de acordo com o tipo de dado (texto, número, url e email) dos termos do bloco.

Para o casamento entre blocos de dados do tipo texto com atributos da base de conhecimento, é utilizado uma função de similaridade chamada de *Attribute Frequency* [Mesquita et al., 2007] que estima a similaridade de um dado valor e um conjunto de valores de um determinado atributo. No caso do ONDUX, essa função calcula a similaridade entre um bloco B e um conjunto de valores de um atributo a, disponível nas ocorrências da base de conhecimento. Para cada atributo é calculado um *score*, o atributo que tiver o maior *score* é eleito como rótulo para esse bloco.

Blocos que contenham apenas números, são tipados como numéricos, e uma função específica de similaridade é usada para esse tipo de dado. Assume-se que os valores numéricos desses dados seguem uma distribuição Gausiana [Chaudhuri and Das, 2003]. Nessa distribuição, é calculada a similaridade entre um valor numérico V_x representado em um bloco B e um conjunto de valores $V(a_i)$ de um atributo a_i presente na base de

conhecimento, avaliando o quão V_x é proximo da média desses valores de $V(a_i)$.

Para blocos do tipo *URL* e *email* é utilizado um pequeno conjunto de funções binárias que utilizam expressões regulares para identificar cada formato específico dos termos dos blocos. Essas funções retornam verdadeiro ou falso de acordo com o casamento desses tipos de dados com os termos do bloco.

Os experimentos desse método aplicado em diferentes domínio, mostram que nas duas etapas citadas (*Blocking* e *Matching*), 80% dos blocos são corretamente rotulados com um nome de atributo. No final da etapa *Matching*, podem ocorrer problemas de rotulação incorreta de blocos, esse problema é decorrente de atributos distintos que possuem domínios com grande interseção de valores. Isso significa que certos atributos podem compartilhar valores e tornar a rotulação ambígua. Quando esse erro ocorre, ele é chamado de *Mismatching*.

A última etapa, chamada de *Reinforcement*, consiste em revisar o pré-rotulamento realizado pela etapa anterior (*Matching*) nos blocos. Nesta etapa, blocos que possívelmente receberam rótulos incorretos (*Mismatching*) são corrigidos, e os quais ainda não receberam um nome de atributo, são rotulados. Para esse reforço os blocos recebem rótulos considerando a posição e sequência dos blocos rotulados no conjunto de texto de entrada a partir de um modelo aprendido.

O modelo aprendido é representado por um grafo chamado de PSM (*Position and Sequencing Model*) parecido com um modelo HMM, ilustramos um exemplo de PSM na Figura 2.5. Este modelo, captura a probabilidade de cada bloco i rotulado com um atributo l aparecer na posição p do texto de entrada, e a probabilidade de cada bloco rotulado por l aparecer antes do bloco rotulado com m no texto de entrada. Estas probabilidades são usadas para fazer o reforço da rotulação dos blocos da etapa anterior(*Matching*), no qual blocos sem rótulos receberão um nome de atributo e blocos que tiveram problema de *Mistmatching*, terão seus rótulos trocados por um nome de atributo correto.

O PSM representa um modelo que consiste em um grafo, no qual representa a posição e sequência dos atributos no texto. Os nós representam os atributos, e as arestas representam a probabilidade de transição entre eles. Seja um conjunto de estados L = {*início*,l_1,l_2,...,l_n,*fim*} no qual, cada estado l_1 corresponde a um atributo atribuído a um bloco na etapa *Matching*, com exceção dos estados *início* e *fim*. Uma Matriz T guarda a probabilidade de uma transição do estado l_i para o estado l_j, e uma Matriz P que guarda a probabilidade do rótulo de l_i de um bloco na posição k da entrada de texto.

Com as matrizes T e P é possível criar o PSM. A matriz T, representando as transições entre os atributos, e a matriz P representa as probabilidades de um atributo

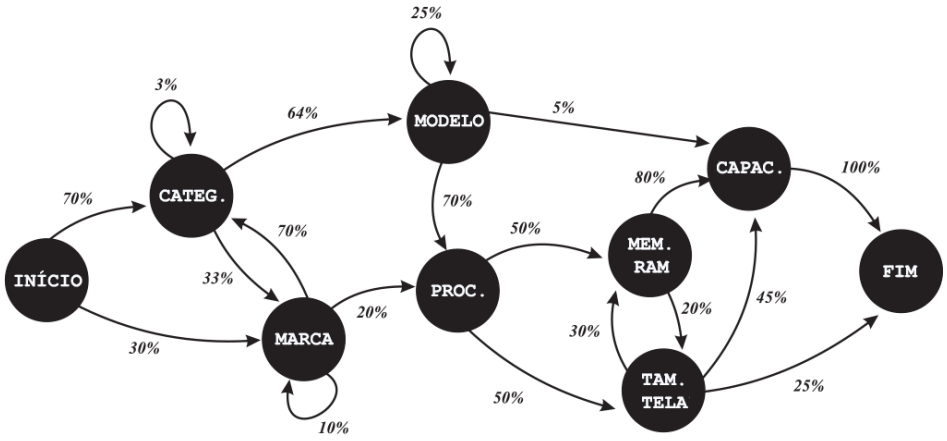

Figura 2.5. Exemplo de PSM(Positioning e Sequence Model)

qualquer aparecer antes ou depois de outro (posição). Para finalizar a etapa de reforço, é combinado a respectiva função de similaridade de cada bloco com as matrizes de estados e transições, T e P. Essa combinação será realizada para bloco da entrada de texto, e para cada atributo da base de conhecimento. A combinação que tiver maior *score*, terá o nome do atributo como rótulo do bloco. Por fim, todo os blocos recebem um rótulo. Os que foram erroneamente rotulado, são corrigidos, e os que não tiveram rótulos, recebem um nome de atributo a partir do modelo PSM.

Como nosso método depende apenas do conteúdo textual das páginas Web, o processo de extração torna-se mais robusto e flexível, tolerante a qualquer problema relacionado as técnicas de extração de dados por alinhamento de árvores. Além disso, não necessitamos que os dados possuam árvores similares, delimitadores explícitos ou qualquer outra informação de estrutura provenientes da árvore DOM das páginas Web, pois utilizamos apenas o conteúdo textual destas páginas. Nosso método, utiliza como base da extração, o método ONDUX que necessita apenas dos registros como entrada. O funcionamento do nósso método será apresentado em maiores detalhes no Capítulo 3.6.

Capítulo 3

FIEX

No capítulo anterior, descrevemos métodos recentes da literatura para extração de dados em páginas Web. A maioria desses métodos combinam técnicas que tentam explorar a estrutura HTML das páginas para encontrar as regiões ricas em dados e extraí-las em um conjunto limitado de cenários: elementos textuais atômicos e elementos textuais compostos. Nesse capítulo, apresentamos nosso método, chamado FIEX (**F**ormat **I**ndependent WebData **Ex**traction) que é capaz de lidar com todos tipos de elementos textuais, inclusive nos casos severos de elementos textuais compostos, nos quais métodos baseados em técnicas de alinhamento de dados e outras similares, não conseguem realizar a extração. De forma diferente dos métodos descritos no Capítulo 2, o FIEX não utiliza a estrutura HTML das páginas para auxiliar na extração, por isso, consideramos nosso método independente de formato.

O FIEX pode ser classificado como um método de extração de dados por segmentação de texto (Information Extraction by Text Segmentation, IETS [Cortez and da Silva, 2013]), e utiliza como base da extração o método ONDUX [Cortez et al., 2010a]. Os métodos atuais de IETS [Cortez et al., 2010a, Cortez et al., 2011, Lafferty et al., 2001b, Borkar et al., 2001], não estão preparados para extrair dados diretamente de páginas Web. A entrada desses métodos consiste em um conjunto de registros que serão segmentados, e posteriormente para cada segmento, será desginado um atributo. O FIEX, ao contrário, recebe como entrada páginas Web.

Como um exemplo de aplicação do FIEX, considere a tarefa de extração de registros de produtos das páginas Web de lojas de comércio eletrônico e seus respectivos dados de forma automática. Registros de produtos são representados pelo conjunto de informações que constituem uma oferta ou descrição de um produto. Ilustramos exemplos deste tipo de registro na Figura 3.1, regiões (a), (b), (c), onde observamos

casos severos de elementos textuais compostos. O valores de atributos representam os dados dos registros que queremos extrair, compostos por um conjunto de valores de atributos sem a presença de delimitadores explícitos ou uma estrutura aninhada de *tags*. Exemplos dessa extração de dados são ilustradas na Figura 3.2, onde valores de 8 atributos distintos estão representados no mesmo elemento HTML, conforme enfatizamos nas regiões (a), (b) e (c) destacadas na Figura 3.1. É importante reenfatizar que os métodos baseados em alinhamento de dados descritos no Capítulo 2, e que são comumente empregrados para este tipo de tarefa, seriam incapazes de lidar com este tipo de página.

Em linhas gerais, a operação do FIEX consiste em 3 fases. Na primeira fase, removemos toda marcação HTML das páginas Web. Na segunda fase, a partir do conteúdo textual restante, conseguimos encontrar a região desse texto que contém os dados de interesse. Para isso, utilizamos uma técnica de eliminação de dados que consiste em remover conteúdos redundantes das páginas [Gulhane et al., 2010]. E, por fim, a partir desta região de dados, extraímos registros utilizando as etapas do método ONDUX (*Blocking/Matching/Reinforcement*). Essas etapas são utilizadas para encontrar, segmentar e rotular registros. **para** q **faça**

u

fim

e o FIEX receba como entrada, páginas WEB ao invés de registros, utilizamos um novo algoritmo para criação de blocos da etapa de *Blocking*. Na etapa de *Matching* tivemos que adaptá-lo para encontrar o início e fim de cada registro das páginas WEB. Estas adaptações do método ONDUX serão detalhadas nas seções que mostram o funcionamento da nossa abordagem.

Nas próximas seções, apresentamos uma visão geral do cenário que nós consideramos para extração e o detalhamento do funcionamento do FIEX.

3.1 Cenário para extração

Em nosso método, consideramos um cenário extremamente comum na estrutura de páginas Web rica em dados, tais como páginas de sites de comércio eletrônico. A caracterização que definimos deste cenário é baseada na larga experiência profissional do aluno na empresa Neemu.com[1] em desenvolvimento de sistemas para extração de dados de páginas Web ricas em dados. Na Figura 3.3 ilustramos a estrutura típica deste tipo de página.

[1] A Neemu.com é uma empresa que fornece serviços de tecnologia para ecommerce. Atualmente, 30% das transações do ecommerce brasileiro passam pela Neemu.com.

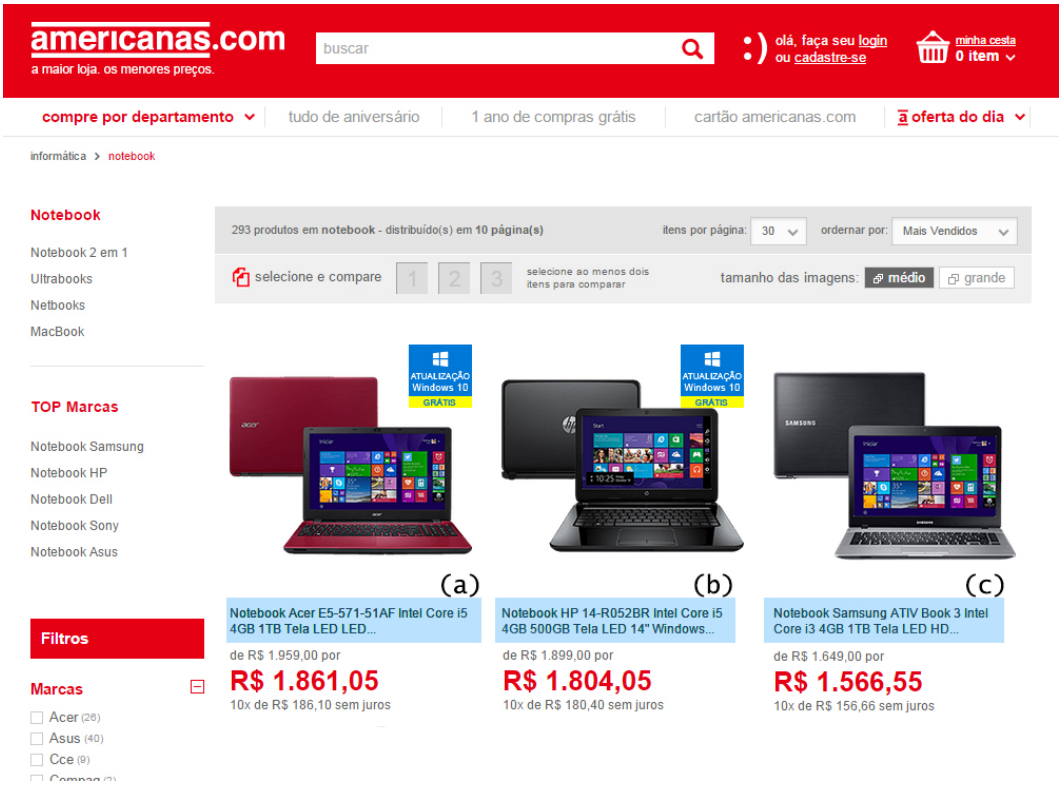

Figura 3.1. Exemplo de registro de produtos(Notebook).

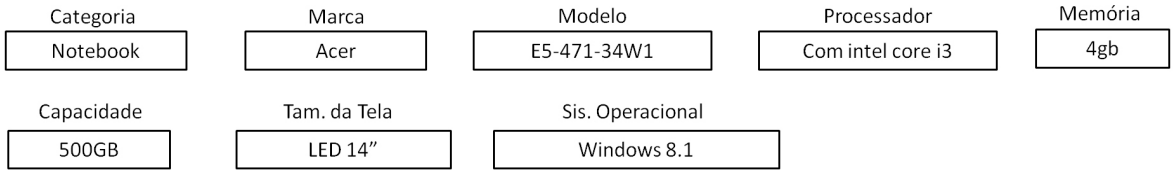

Figura 3.2. Exemplo de Extração de dados em registros(Notebook).

No cabeçalho destas páginas, existem várias informações irrelevantes (menus, links, banners, etc), responsáveis pela navegação do usuário pela página. Nas regiões laterais destas páginas, temos um conjunto de filtros comuns para cada domínio procurado, além de banners e propagandas. No rodapé, encontramos uma região com informações sobre a loja, descrição da loja, telefones e *emails* para contato, etc. Na região central destas páginas, é possível identificar um conteúdo dinâmico, composto por registros de produtos e seus repectivos dados que mudam de acordo com a interação que o usuário faz na página. Essa região é a que contém os dados que deverão ser extraídos.

A existência deste *layout* típico possibilita remover conteúdos desnecessários a

partir de dados redundantes de um conjunto de páginas Web. Os dados redundantes são normalmente representados por menus, links de propagandas, banners, links de navegação e qualquer outra informação que não é considerada como um registro de produto. Esses dados sempre estão dispostos nas regiões laterais, no cabeçalho e rodapé das páginas. Desta forma, facilmente encontramos a região rica em registros para prosseguir com a extração de dados. Em nossa abordagem, exploramos essa disposição dos dados para eliminar conteúdos redundantes, e encontramos de forma automática blocos de informação que contenham registros e seus respectivos dados para extração. Nas páginas de 10 lojas utilizadas nos experimentos, foi possivel confirmar essa hipótese, pois conseguimos de forma eficiente encontrar os registros dos documentos e extrair os dados. Mais detalhes sobre a eficiência da extração, serão apresentadas no Capítulo 4, que mostra nossos experimentos.

Figura 3.3. Exemplo da disposição dos dados em páginas Web de um comércio eletrônico

3.2 Visão geral

Nesta seção, apresentamos uma visão geral do nosso método, cuja descrição em alto nível é apresentada no Algoritmo 1. O FIEX recebe como entrada um conjunto de páginas Web de um mesmo site e tipo de produto, produzindo como saída um conjunto de arquivos XML com a representação dos dados extraídos. Para cada página de entrada, executamos 3 fases. Na primeira fase, Linhas 3-5 do Algoritmo 2, removemos todas marcações HTML das páginas Web. Na segunda fase, Linha 6 do algoritmo, a partir do conteúdo textual restante da primeira fase, conseguimos encontrar a região do texto que contém os dados de interesse. Para isso, utilizamos uma técnica de eliminação de dados que consiste em remover conteúdos redundantes das páginas [Gulhane et al., 2010]. E, por fim, nas Linhas 7-11, a partir desta região de dados, extraímos os registros utilizando as fases de *Blocking/Matching/Reinforcement* do ONDUX [Cortez et al., 2010a]. Essas fases são utilizadas para encontrar, segmentar e rotular registros. As adaptações do método ONDUX serão detalhadas nas seções que mostram o funcionamento das fases do FIEX.

1: **Entrada** : $A = \{a_1, \ldots, a_n\}$ {Conjunto de páginas Web}
2: **Saída** : $X = \{x_1, \ldots, x_n\}$ {Conjunto de arquivos XML com os dados extraídos}
3: **for all** a_i em A **do**
4: $h_i \leftarrow$ Remove HTML de a_i; {Seção 3.3}
5: **end for**
6: $L = \{\ell_1, \ldots, \ell_n\} \leftarrow$ Remove Conteúdo Redundante de $h_1, \ldots, h_n$; {Seção 3.4}
7: **for all** ℓ_i em L **do**
8: $b_i \leftarrow$ Blocking(ℓ_i); {Seção 3.5.1}
9: $m_i \leftarrow$ Matching(b_i); {Seção 3.5.2}
10: $x_i \leftarrow$ Reiforcement(m_i); {Seção 3.5.3}
11: **end for**
12: **return** $X = \{x_1, \ldots, x_n\}$

Algorithm 1: Funcionamento FIEX

Nas Seções 3.3, 3.4 e 3.5 apresentamos respectivamente o funcionamento detalhado da nossa abordagem, iniciando pela remoção do HTML, depois a eliminação do conteúdo redundante e por fim, apresentamos a extração dos registros e seus respectivos dados.

3.3 Remoção da marcação HTML

A primeira etapa do nosso método, consiste em remover das páginas Web qualquer elemento que não seja texto (tags HTML, CSS, links de navegação, banners, imagens,

etc).

Isto é feito porque em alguns casos de elementos textuais compostos e em todos casos severos de elementos textuais compostos, é comum que a representação da estrutura lógica das páginas Web comparada com a estrutura conceitual dos dados nelas representadas seja inconsistente. A estrutura lógica, é representada pela Árvore DOM destas páginas cuja estrutura é aplicada pelas técnicas de extração tradicionais, como por exemplo, de alinhamento de dados. A estrutura conceitual é representada pelos atributos dos dados de interesse.

Em aplicações como apresentação de produtos, que contém casos severos de elementos textuais compostos, não é possível associar a estrutura lógica com a extração final dos dados, pois existem vários valores de atributos para serem extraídos sendo representados por apenas um elemento HTML. Esta inconsistência entre a representação da estrutura dos elementos da Árvore DOM e a estrutura dos dados de extração, impossibilita os métodos baseados em alinhamento de dados de extraí-los. Sendo assim, não é possível confiar na estrutura, e removemos todas marcações HTML destas páginas.

Para esta remoção, utilizamos a Jericho [Parser, b]. Jericho é uma biblioteca em JAVA para manipulação de páginas Web que possui uma série de funções para realizar tarefas específicas em páginas Web de forma eficiente. Nela utilizamos uma função de limpeza que auxilia a tarefa de remoção desse conteúdo não textual. A entrada dessa função consiste em uma página Web completa, e após a execução da função, temos como saída, apenas as informações que estão entre as tags HTML, ou seja, o texto.

Para esta etapa, até a utilização de expressões regulares poderia ser considerada válida. Entretanto, existem problemas em utilizar abordagens triviais, pois muitos scripts inseridos diretamente no HTML ou erros de programação podem remover incorretamente as *tags*, adicionando ruído ao conteúdo textual que prejudica a extração final dos dados. Por isso, um removedor de *tags* HTML eficiente é importante nessa etapa.

Testamos algumas bibliotecas para manipulação de páginas Web (JSOUP [jsoup Java HTML Parser,], HTMLParser [Parser, a] e Jericho HTML Parser [Parser, b]. Estas bibliotecas foram testadas em um conjunto de 20 páginas Web diferentes, previamente coletadas de lojas de comércios eletrônicos contendo descrições de produtos. A JSOUP e o HTMLParser não conseguiram remover toda as marcações HTML, pois junto com o conteúdo textual foram erroneamente deixados outros conteúdos que são referentes à linguagem HTML e outros elementos que compõem essas páginas. Além disso, as implementações dessas duas bibliotecas são fechadas, dificultado alguma necessidade de adaptação do código. O JSOUP e o

HTML Parser conseguiram extrair parte textual da página HTML de forma correta sem a presença de qualquer parte da linguagem HTML. A biblioteca que obteve melhor resultado foi a *Jericho*, pois eliminou corretamente todos elementos não pertencentes ao conteúdo textual desse conjunto de teste. Na Figura 3.4, ilustramos o processo de remoção de conteúdo não textual. Na região (a) desta figura, temos duas páginas Web com suas marcações HTML e em (b) o resultado da remoção do HTML pelo Jericho.

3.4 Extração de conteúdo relevante

Após remover todas informações não textuais, é necessário encontrar a região do texto que contém apenas registros. Quando removemos informações estruturais das páginas, o conteúdo textual de links de navegação, como menus laterais, filtros e propagandas ainda estão presentes e misturados com os dados relevantes à extração final. Desta forma, para eliminar esse conteúdo irrelevante, utilizamos uma técnica de eliminação de conteúdo baseada em redundância da informação [Gulhane et al., 2010]. Essa técnica consiste em remover conteúdos que se repetem nos documentos, realizando uma filtragem que extrai apenas a região que contém os registros dos produtos. Para que isto funcione, é necessário ter um conjunto de páginas Web de uma mesma loja e de um mesmo tipo de produto.

Para auxiliar essa tarefa, utilizamos o *DIFF*, que é uma ferramenta presente em repositórios das principais distribuições Linux. Seu processo é simples, e consiste em comparar o conteúdo de dois arquivos e mostrar a diferença entre eles. Na ferramenta, é possível ajustar configurações de impressão da saída de dados e redirecionar as diferenças encontradas para um novo arquivo. Em nosso caso, o DIFF foi ajustado para receber dois arquivos como entrada e retornar apenas o conteúdo diferente do primeiro arquivo em relação ao segundo, para isso utilizamos duas configurações (changed-group-format=%> e unchanged-group-format=“ ”).

Desta forma, considere o conjunto H=$\{h_1, ..., h_n\}$, onde cada h_i é um arquivo de texto resultante da remoção de marcação HTML (Ver seção 3.3). A extração do conteúdo relevante, consiste em selecionar aleatóriamente um destes textos como padrão, t_1, e para cada elemento do conjunto H, utilizamos o *DIFF* com as páginas restantes deste conjunto. O *DIFF* recebe dois arquivos como entrada: o padrão, t_1 e um texto h_i do conjunto H. Após isso, são feitas comparações entre os arquivos, e com a diferença do texto h_i em relação ao t_1 é criado um novo arquivo. Essa diferença, representa uma sequência de textos que estão em h_i e não estão em t_1, ou seja, textos redundantes entre os arquivos são removidos. Um exemplo é ilustrado na Figura 3.5. Nesta figura,

Figura 3.4. Remoção das marcações HTML com a Jericho(b1 e b2) e resultado do DIFF (c)

realizamos o DIFF visual com duas páginas diferentes. A região lateral, o cabeçalho e o rodapé são conteúdos redundantes, desta forma, essas regiões são eliminadas. A região central é a única restante, nela estão contidos os dados dinâmicos (não redundantes) que serão utilizados na extração. Um outro exemplo do DIFF, mas aplicado diretamente no conteúdo textual das páginas está ilustrado na Figura 3.4, regiões b_1 e b_2. Nestas regiões, temos em destaque na cor vermelha o conteúdo textual reduntante que se repete nas duas páginas de entrada. Os textos na cor azul representam o conteúdo dinâmico, e na região (c) da figura, temos o resultado do *DIFF* que contém somente os dados de interesse para extração.

Existe um problema no processamento do *DIFF*, o padrão t_1 não pode ser comparado com ele mesmo, uma vez que seria gerado um arquivo vazio pois os documentos para comparação e extração são idênticos. Para contornar esse problema, elegemos qualquer outro elemento do conjunto H diferente de t_1. Desta forma, essa novo arquivo padrão t_2 é comparada com o arquivo padrão anterior t_1 e é gerado um novo arquivo para o conjunto L. O processo desta etapa está apresentado no Algoritmo 2.

No final desse processo, temos um novo conjunto de arquivos L=$\{l_0, ..., l_n\}$ cada um com um conjuntos de termos que representam os dados e compões os registros

1: **Entrada** : $C = \{C_0, ..., C_n\}$ {Conjunto de páginas de um tipo D e loja L}
2: **Saída** : $L = \{L_0, ..., L_n\}$ {Conjunto de arquivos com dados de interesse}
3: j=0;
4: $T_1 \leftarrow TemplateAleatorio(C)$;
5: **for all** C_i em C **do**
6: **if** $C_i == T_1$ **then**
7: $T_2 \leftarrow TemplateAleatorio(C - T_1)$;
8: $L_j = DIFF(T_2, T_1)$;
9: **else**
10: $L_j = DIFF(C_i, T_1)$;
11: **end if**
12: j++;
13: **end for**

Algorithm 2: Eliminação de conteúdo redundante

Na próxima etapa do FIEX, que está descrita na próxima seção, os dados de cada arquivo do conjunto L serão segmentados em registros, e a partir desses registros, serão extraídos os dados.

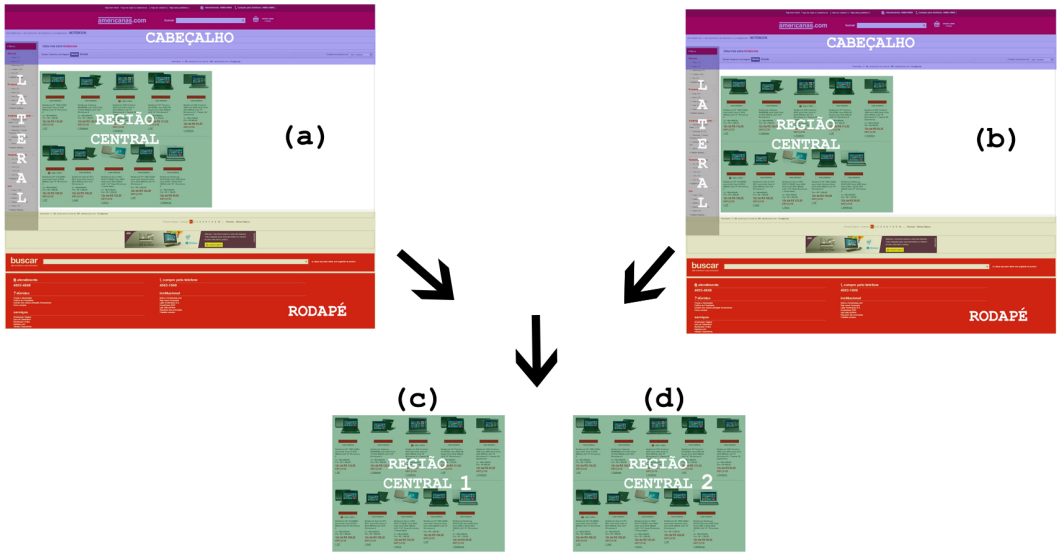

Figura 3.5. Exemplo do algoritmo DIFF em duas páginas Web

3.5 Extração automática de registros e dados

Nesta etapa, após encontrar a região dos dados, é necessário segmentar esses dados e encontrar os registros que compõem essas regiões. Para isso, utilizamos os passos de *Blocking* seguida do passo de *Matching* de forma similar ao que é feito no ONDUX.

3.5.1 Blocking

Neste passo, segmentamos em blocos o texto de cada arquivo do conjunto L, gerado na etapa anterior do FIEX. Para cada arquivo, criamos um conjunto de blocos, que são simples sequências de termos (palavras) que são susceptíveis de formar um valor de um atributo.

Como descrito na Seção 2.3 do Capítulo 2, no ONDUX, a criação de blocos é feita com base na simples co-ocorrência de termos na base de conhecimento. Em nosso cenário, no entanto, essa estratégia se mostrou pouco efetiva, razão pela qual, desenvolvemos um algoritmo alternativo que combina a ocorrência de n-gramas, com a estratégia original baseada em termos simples. Esta adaptação foi importante devido à estrutura composta dos valores que representam um atributo, isto quer dizer que para um único atributo podemos ter até três termos que o representa. A abordagem anterior de blocagem apresentou problemas por considerar que cada termo que compõe o atributo pode ser considerado inicialmente como um bloco. Na etapa de *Matching* do ONDUX, é realizada a rotulação destes blocos, neste momento, existe a possibilidade de um desses termos estar presente em algum outro atributo, ou seja, outro

bloco criando a chance de algum desses termos ser rotulado de forma incorreta. Em nossa abordagem baseada em NGRAM, a idéia é reduzir este erro na rotulação quando consideramos que um conjunto de termos representem apenas um bloco, e não vários termos representando vários blocos.

Este algoritmo recebe como entrada uma sequência T de n termos retirados de um arquivo de texto. Para esta sequência de termos, inicialmente são gerados n-gramas de tamanho 3. Para cada n-grama gerado, verificamos a ocorrência deste n-grama na base de conhecimento. Se o n-grama existir nesta base, é criado um bloco conténdo os termos deste n-grama. Além disso, os termos do n-grama encontrados são removidos da entrada T de termos. Após a verificação dos n-gramas de tamanho 3, são gerados novos n-gramas de tamanho 2 com os termos restantes de T. O mesmo processo de verificação e remoção feito anteriormente é realizado para estes n-gramas. Após esta última verificação, os termos em T que restaram, são todos de tamanho 1, sendo assim, cada um desses valores irá representar um novo bloco. Na Figura 3.6 região (a), ilustramos uma sequência T de termos de entrada e na região (b) apresentamos o resultado do passo de *Blocking* aplicado nesta sequência.

3.5.2 Matching

Após a segmentação do texto em blocos, executamos o algoritmo *Matching* que irá atribuir rótulos a esses blocos. Nesta fase, os blocos são comparados com os exemplos de valores conhecidos dos atributos (base de conhecimento), para isso, a etapa *Matching* do ONDUX utiliza um pequeno conjunto de funções de similaride específicas para a correspondência entre blocos e atributos da base. Ao final da fase de *Matching*, temos uma sequência blocos, na qual, cada bloco é pré-rotulado com o nome de um atributo que teve a melhor similaridade entre atributos da base e os termos que compõe o bloco.

A partir desta sequência pré-rotulada de blocos, é possível encontrar registros implícitos de descrições de produtos. Para isso, adaptamos a etapa *Matching* do ONDUX para identificar esses registros segmentando a sequência a partir de uma premissa, na qual, todo início de registro de uma descrição de produto contém informação sobre tipos de produtos (Notebook, TV, Smartphones, etc.) ou a marca (Sony, Samsung, etc.) dele.

Isto ocorre na grande maioria dos registros de produtos, pois conseguimos extraí-los de forma efetiva, como demonstrado em nossos resultados experimentais que serão detalhados no próximo capítulo.

Desta forma, dado uma sequência rotulada de blocos B, a cada atributo de Tipo de produto ou Marca encontrado, todos os blocos vizinhos a esse atributo irão compor

um registro até que outro bloco com um destes atributos seja encontrado. Sendo assim, separamos os registros de descrições de produtos, utilizando apenas os dados rotulados na fase de *Matching*.

Na Figura 3.6, região (c), ilustramos esta fase que atribui rótulos para os blocos e a segmentação da sequência de blocos em registros. Ainda nesta região, é possível observar que os 3 registros apresentados iniciam com o atributo "Tipo de produto", conforme a estrutura típica dos registros que mencionamos anteriormente.

A segmentação da sequência de blocos em registros, também pode ser realizada sem a necessidade de conhecer a natureza dos registros. Para isto, é possível utilizar métodos de extração de informação por segmentação de texto [Cortez et al., 2011] que são aplicados especificamente para esses casos. Com estes métodos, é possível extrair os registros de uma sequência rotulada de blocos sem conhecer sua estrutura. Nesse trabalho, ainda não utilizamos esses métodos, deixamos esta possibilidade como trabalho futuro.

Como descrito em [Cortez et al., 2010b], as etapas de *Blocking* e *Matching* do ONDUX são passos suficientes para rotular corretamente a grande maioria dos segmentos na entrada. Os experimentos com diferentes bases de conhecimentos mostram que blocos estão corretamente pré-rotulados em mais de 80% dos casos. Para os blocos que ainda não foram rotulados ou que possuam rótulos errados, é realizada a fase de reforço (*Reinforcement*), que será detalhada a seguir..

3.5.3 Reinforcement

A fase *Reinforcement* [Cortez et al., 2010b] complementa as fases de *Matching* e *Blocking* realizadas anteriormente. O *Reinforcement* é baseado principalmente em um modelo conhecido como PSM (Positioning and Sequencing Model), que reconhece a posição e sequência dos atributos dos registros. Com este modelo, é possível adicionar rótulos aos blocos que ainda não foram rotulados, e corrigir blocos que receberam nomes de atributos de forma incorreta (*Mismatching*).

Na região (c) da Figura 3.6, ilustramos alguns blocos com nomes de atributos na cor vermelha, estes blocos foram erroneamente rotulados ou não tiveram seus rótulos atribuídos a partir da fase de *Matching*. Em seguida, na região (d) desta figura, e após o processo da fase de *Reinforcement*, estes blocos estão com seus atributos na cor azul que representa as correções da etapa de reforço.

Com a extração dos dados finalizada, é possível estruturá-la em alguma tabela ou arquivo XML. Em nosso caso, a extração final dos dados consiste em um conjunto

de arquivos XML que contém os atributos (*tags*) e os dados (valores das *tags*) de cada registro (linha do arquivo) de cada página de entrada.

Todos os passos realizados em sequência, estão representado pelas regiões (a), (b), (c) e (d) da Figura 3.6.

No próximo capítulo, apresentamos nossos experimentos para mostrar que a nossa abordagem baseada apenas em informação textual, é muito eficiente para extração de dados em páginas Web contendo descrições de produtos.

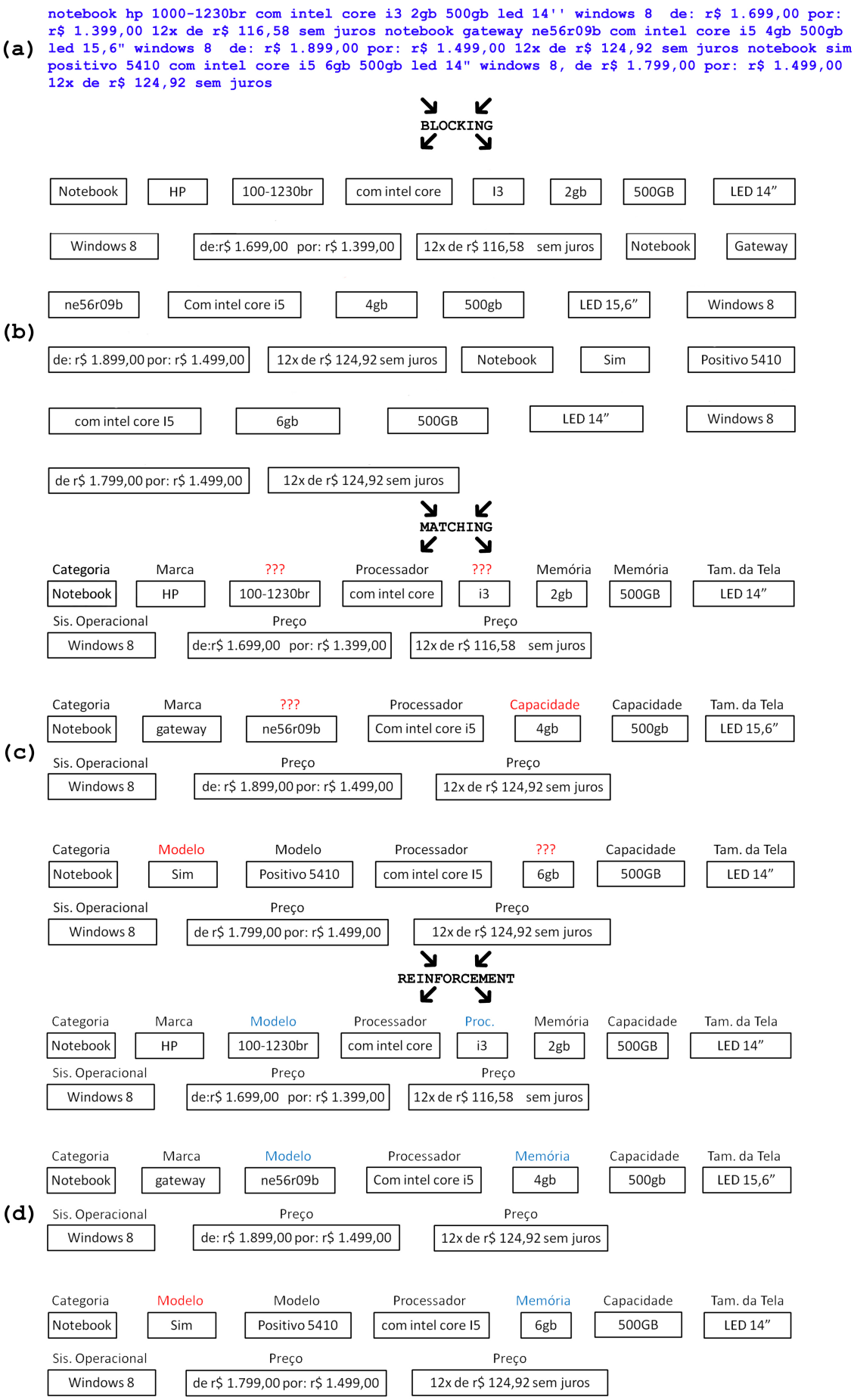

Figura 3.6. Exemplo de extração de registros e dados

Capítulo 4

Resultados Experimentais

Neste capítulo, apresentamos os resultados experimentais obtidos pelo FIEX na extração de dados de descrição de produtos em páginas Web de comércio eletrônico. Primeiramente, descrevemos o conjunto de métricas adotadas para medir a eficácia do FIEX. Em seguida, apresentamos os dados de experimentos que descrevem as páginas com os tipos diferentes produtos e seus respectivos atributos de extração utilizados nestes experimentos. Por fim, apresentamos os resultados da qualidade da extração dos dados de descrição de produtos utilizando o FIEX.

Nos experimentos, avaliamos o resultado da extração obtido em duas etapas importantes do FIEX. Primeiro, na extração dos registros das páginas Web, e segundo, na extração de dados de cada registro que têm seus termos rotulados com um respectivo atributo (em nível de atributo e registro).

4.1 Métricas Utilizadas

Para avaliar os resultados obtidos dos nossos experimentos em nível de atributo e registro, utilizamos métricas conhecidas na literatura de recuperação de informação [Baeza-Yates and Ribeiro-Neto, 2011] Precisão, Revocação e Medida F. Estas métricas serão definidas a seguir.

A precisão calcula a quantidade de respostas corretas em relação ao total de respostas retornadas. Está métrica é representada pela fórmula da Equação 4.1. A Revocação é a quantidade de respostas corretas em relação ao total de respostas esperadas ou relevantes, cuja fórmula está representada na Equação 4.2. A Medida F, é definida como a média harmônica da Precisão e da Revocação, cuja fórmula está representada na Equação 4.3. A aplicação destas métricas na avaliação do experimentos

será detalhada nas próximas seções.

$$Precisao = \frac{|\{respostas\ relevantes\} \cap \{respostas\ retornadas\}|}{|\{respostas\ retornadas\}|} \tag{4.1}$$

$$Revocaco = \frac{|\{respostas\ relevantes\} \cap \{respostas\ retornadas\}|}{|\{respostas\ relevantes\}|} \tag{4.2}$$

$$MedidaF = \frac{2(Revocacao \times Precisao)}{(Revocacao + Precisao)} \tag{4.3}$$

4.2 Base de dados dos experimentos

Em nosso trabalho, apresentamos uma solução para extração de dados em páginas Web contendo casos severos de elementos textuais compostos. A maioria dos métodos apresentados no Capítulo 2, não conseguem extrair os dados deste tipo de caso. Além disso, nos trabalhos de extração de dados em páginas Web, existe uma ausência de coleções de referência amplamente aceita e difundida. Por isso, para avaliar o método FIEX, procuramos selecionar páginas Web de comércio eletrônico que contenham um conjunto representativo de descrições de produtos.

A base de dados dos nossos experimentos, consiste em um conjunto de páginas Web coletadas das 14 lojas mais conhecidas de comércio eletrônico: Americanas, Casas Bahia, Colombo, Extra, Girafa, Kabum, LojasMM, Magazine Luiza, MegaMamute, Ponto Frio, Saraiva, Shoptime, Submarino e Walmart. Estas lojas possuem vários tipos de produtos: Notebooks, Televisores, Celulares, Jogos, Eletrodomésticos, etc. Para cada loja, foram coletados aproximadamente 8 páginas Web para cada tipo de produto, os quais contêm, em média, 20 registros por página. Realizamos os experimentos em três desses tipos diferentes desses produtos com diferentes atributos para extração. Na Tabela 4.1, apresentamos os tipos de produtos e seus respectivos atributos utilizados para avaliar a extração de dados do FIEX em páginas Web.

As bases de conhecimento, foram selecionadas a partir do banco de dados da empresa que presta serviços para sites de comércio eletrônico [Neemu.com,], Esse banco contém exemplos de atributos e seus valores para produtos de Notebook, TV e Smartphones. Para nossos experimentos utilizamos bases de conhecimento com 8975 termos que representam 3752 exemplos de atributos e valores distribuídos entre as os produtos de Notebook, TV e Smartphones.

Domínios	Atributos
Notebook	[Processador, Preço, Tipo de produto, Modelo da Marca, Marca, Sistema Operacional, HD, Memória, Tamanho da tela, Mais informações]
TV	[Modelo, Marca, Tamanho da tela, Frequência, Preço, Cor, Tipo de produto, Entradas, Tipo da Tela, Mais informações]
Smartphones	[Tipo de produto, Marca, Capacidade, Tamanho Tela, Memória, Processador, Câmera, Mais informações]

Tabela 4.1. Tabela com atributos extraídos de cada domínio

4.3 Metodologia de Avaliação

Para avaliação dos resultados obtidos pelo FIEX, foram selecionados por amostragem 20 registros em média para cada loja.

É considerado como registro, um conjunto de dados que contém nome do produto, atributos do produto e preço. As quantidades de páginas e atributos avaliados serão descritos futuramente na Seção 4.4, que descreve a avaliação dos resultados em nível de registro e de atributo.

A avaliação dos resultados foi dividida em três partes brevemente descritas a seguir e detalhadas nas Subseções 4.4.1, 4.4.2 e 4.4.3.

Na primeira, chamada de Avaliação da Extração de Registros, inicialmente selecionamos uma página aleatória de cada loja e de cada tipo de produto. Esta página que está em formato HTML, contém os registros originais da página. Depois isso, manualmente comparamos a quantidade registros extraídos pela segunda etapa do FIEX nesta página e a página selecionada inicialmente.

Na segunda, chamada de Avaliação por Atributo dos Registros Rotulados, avaliamos a extração completa do FIEX que tem como resultado à atribuição de rótulos para os registros extraídos. Para isso, foi criado uma interface conténdo o resultado da extração do FIEX. Nela, foram apresentandos alguns blocos de dados que compõem registros com seus respectivos rótulos. Um total de 40 usuários avaliaram se a rotulação do bloco estava correta ou incorreta. Esses dados tem como objetivo mostrar a eficácia da extração do FIEX. Mais detalhes sobre essa avaliação, estão descritas na subseção 4.4.2.

Na última, chamada de Avaliação por Registro das Lojas, é apresentado uma avaliação dos resultados da extração de cada loja. Foram selecionados de forma alea-

tória alguns registros e blocos rotulados das páginas extraídas de cada loja. O mesmo processo de avaliação anterior foi utilizado para esta avaliação. No total temos 144 registros e 3588 termos com seus respectivos atributos para o tipos de produtos de Notebooks, 181 registros com 4681 termos com seus respectivos rótulos para TVs e 132 registros com 1552 termos com seus respectivos rótulos para tipos de produtos de Smartphones. O cálculo da precisão da extração de um registro considera, respectivamente, a média aritmética da precisão da extração dos valores dos atributos deste registro. Em outras palavras, a precisão da extração de um registro será dada pela média dos valores de precisão da extração de todos os seus atributos.

4.4 Experimentos da extração

Nesta seção são apresentados e discutidos os resultados obtidos pela extração do FIEX nas coleções de páginas apresentadas anteriormente e em cada tipo de avaliação. Para estas três avaliações brevemente descritas na seção anterior, apresentamos tabelas que contém valores para cada tipo de avaliação utilizando as métricas de Precisão, Revocação e Medida F. As tabelas que contém informações sobre a extração dos dados apresentam apenas a métrica de precisão.

4.4.1 Avaliação da Extração de Registros

Neste experimento, avaliamos a eficiência das etapas de Remoção de Conteúdo Redundante e Extração de Registro. Para isso, apresentamos nas Tabelas 4.2, 4.3 e 4.4 os resultados obtidos nestas etapas para cada tipo de página de produto e para cada loja da nossa coleção. Nestas tabelas, a coluna "Relevantes", se refere à quantidade de registros que a página HTML original possui. Esse valor é obtido de forma manual na qual uma pessoa conta quantos registros existem nesta página. Este valor é considerado como o gabarito. A coluna "Extraídos", destas tabelas, representa a quantidade de registros extraídos pelo FIEX. Essa quantidade também foi contabilizada de forma manual.

As Tabelas 4.2, 4.3 e 4.4 na grande maioria das lojas, mostram que os registros foram extraídos de forma correta conforme a média de Medida F apresentada. Nessa três tabelas, as siglas Rel.,Ext.,Inc.,Cor.,P.,R.,MF. representam registros Relevantes, Extraídos, Incorretos, Corretos, Precisão, Revocação e Medida F, respectivamente.

É perceptível que em algumas lojas como, Lojas MM e Kabum, os valores de precisão podem ser considerados baixos. Isto é uma consequência de um problema da etapa de Remoção de Conteúdo Redundante do FIEX. Por exemplo, quando analisado

Lojas	Rel	Ext.	Inc.	Cor.	P.	R.	MF.
Americanas	28	28	1	27	0.96	0.96	0.96
Casas Bahia	20	21	2	19	0.90	0.95	0.93
Colombo	15	15	0	15	1.00	1.00	1.00
Extra	20	21	1	20	0.95	1.00	0.98
Girafa	12	11	0	11	1.00	0.92	0.96
Kabum	20	33	13	20	0.61	1.00	0.75
LojasMM	20	40	20	20	0.50	1.00	0.67
Magazine Luiza	20	20	0	20	1.00	1.00	1.00
Mamute	20	22	2	20	0.91	1.00	0.95
Ponto Frio	20	20	0	20	1.00	1.00	1.00
Saraiva	10	10	0	10	1.00	1.00	1.00
Shoptime	20	20	0	20	1.00	1.00	1.00
Submarino	20	20	1	19	0.95	0.95	0.95
Walmart	20	15	0	15	1.00	0.75	0.86
					0,91	0,97	0.93

Tabela 4.2. Resultados da extração de registros em produtos de Notebooks

Lojas	Rel	Ext.	Inc.	Cor.	P.	R.	MF.
Americanas	28	39	12	27	0,70	0.96	0.81
Casas Bahia	20	21	1	20	0.95	1.00	0.98
Colombo	15	15	0	15	1.00	1.00	1.00
Extra	20	26	6	20	0.77	1.00	0.87
Girafa	12	14	2	12	0.86	1.00	0.92
Kabum	20	21	1	20	0.95	1.00	0.98
LojasMM	20	42	20	20	0.48	1.00	0.65
Magazine Luiza	40	58	18	40	0.69	1.00	0.82
Mamute	20	27	7	20	0.74	1.00	0.85
Ponto Frio	20	21	1	20	0.95	1.00	0.98
Saraiva	10	10	0	10	1.00	1.00	1.00
Shoptime	13	26	13	13	0.50	1.00	0.67
Submarino	20	20	1	19	0.95	0.95	0.95
Walmart	20	22	9	13	0.59	0.65	0.62
					0.79	0.96	0.86

Tabela 4.3. Resultados da extração de registros em produtos de TV

Lojas	Rel	Ext.	Inc.	Cor.	P.	R.	MF.
Americanas	28	28	0	28	1.00	1.00	1.00
Casas Bahia	20	21	1	20	0.95	1.00	0.98
Colombo	15	18	3	15	0.83	1.00	0.91
Extra	20	20	0	20	1.00	1.00	1.00
Girafa	12	16	4	9	0.56	0.75	0.64
Kabum	20	38	18	17	0.45	0.85	0.59
LojasMM	20	36	16	20	0.56	1.00	0.71
Magazine Luiza	40	40	0	40	1.00	1.00	1.00
Mamute	20	26	6	20	0.77	1.00	0.87
Ponto Frio	20	20	0	20	1.00	1.00	1.00
Saraiva	10	10	0	10	1.00	1.00	1.00
Shoptime	13	13	0	13	1.00	1.00	1.00
Submarino	20	20	0	20	1.00	1.00	1.00
Walmart	20	20	0	17	0.85	0.85	0.85
				Médias	0.855	0.96	0.89

Tabela 4.4. Resultados da extração de registros em produtos de Smartphones

os problemas da baixa precisão na loja Kabum, foi verificado que existe uma breve descrição em texto contínuo dos benefícios do produto para o usuário, essa descrição contém informações textuais irrelevantes que resulta na criação de registros a mais ou incompletos. Além disso, em algumas lojas, existem textos repetidos com partes do nome da oferta do produto em regiões importantes do documento. Sendo assim, a extração do FIEX destes registros acaba sendo duplicada. Este problema não foi tratado neste trabalho, entretanto isso está relacionado como um trabalho futuro pois a solução deste problema resulta no aumento de precisão da extração nesses casos isolados.

4.4.2 Avaliação por atributo

As Tabelas 4.5, 4.6 e 4.7 mostram os resultados obtidos pelo FIEX na etapa de extração de dados dos registros extraídos pelas etapas anteriores. As colunas "Atributos"destas tabelas, representam os atributos utilizados para a extração de dados em cada tipo de produto.

Nestas tabelas, os resultados da métrica de Precisão, confirmam que o FIEX consegue extrair e rotular os dados de forma correta em 86% dos casos para Notebook e 79% TV. Os resultados se mostram interessantes, quando enfatizamos que os métodos baseados em alinhamento de árvores não extraem dados dos casos severos de elementos textuais compostos e que todo o processo de extração realizado pelo FIEX é puramente

Atributos	Corretos	Errados	Total	Precisao
Processador	258	30	288	0.90
Preço	1403	142	1545	0.91
Tipo de produto	156	0	156	1.00
Modelo da Marca	55	18	73	0.75
Marca	252	20	272	0.93
Sistema Operacional	125	9	134	0.93
HD	145	4	149	0.97
Memória	154	5	159	0.97
Tamanho da tela	172	47	219	0.79
Mais informações	307	301	608	0.50
		Médias	3603	0.86

Tabela 4.5. [Resultados da avaliação por atributo de produtos do tipo Notebook

Atributos	Corretos	Errados	Total	Precisao
Preço	2229	231	2460	0.91
Cor	15	0	15	1.00
Tipo de produto	144	36	180	0.80
Entrada	184	555	739	0.25
Modelo	87	13	100	0.87
Marca	245	38	283	0.86
Tipo da Tela	165	35	200	0.82
Frequência	8	0	8	1.00
Resolução	22	16	38	0.58
Tamanho da Tela	157	27	184	0.85
Mais Informações	336	88	424	0.79
		Média	4631	0.79

Tabela 4.6. [Resultados da avaliação por atributo de produtos do tipo TV

baseado no conteúdo textual destas páginas.

Nesta avaliação, foi verificado que o motivo principal de uma precisão menor em algumas lojas, é decorrente da alta irregularidade na descrição das ofertas. Estas ofertas são escritas sem a presença de uma ordem padrão na apresentação dos atributos, com isso, a fase de reforço do método pode confundir rótulos de atributos. Este problema foi notado na avaliação realizada pelos usuários, na qual é marcado o atributo correto ou incorreto. Este tipo de problema, apesar de não ocorrer em todas lojas, afeta o PSM (*Positioning and Sequence Model*) gerado na fase de reforço e, como consequência, pode-se atribuir rótulos incorretos para os blocos. Estudando os registros que foram marcados incorretos pelos usuários, percebe-se que os erros aparecem nos casos em que os registros de uma página possui os atributos em posições diferentes.

Atributos	Corretos	Errados	Total	Precisao
Preço	1303	184	1487	0.88
Cor	18	0	18	1.00
Tipo de produto.txt	128	3	131	0.97
Modelo	89	35	124	0.71
Marca.txt	148	12	160	0.92
Memoria	93	18	111	0.84
Capacidade.txt	119	29	148	0.81
Processador	136	36	172	0.79
Camera	150	18	168	0.89
Tamanho da tela.txt	160	3	163	0.98
Mais informações.txt	90	25	115	0.78
		Média	2797	0.87

Tabela 4.7. [Resultados da avaliação por atributo de produtos do tipo Smartphones

Lojas	Registros	Entidades	Acertos	Erros	Precisão
CasasBahia	20	511	406	105	0.79
Colombo	15	293	237	56	0.81
Extra	20	491	405	86	0.82
FasShop	12	390	355	35	0.91
FNAC	6	94	76	18	0.80
Girafa	12	269	241	28	0.89
Magazine	29	769	586	183	0.76
Mamute	20	355	304	51	0.85
Saraiva	10	281	238	43	0.84
PontoFrio	20	490	387	103	0.78
	144	3588			0.82

Tabela 4.8. [Resultados da avaliação por loja de produtos do tipo Notebook

4.4.3 Avaliação por registro das Lojas

Nesta avaliação, apresentamos de uma forma unificada, os resultados da extração de dados da avaliação anterior. Desta forma, as avaliações de todos os atributos, são unificados em uma única unidade representada pela loja. Os dados desta tabela, demonstram que nosso método é capaz de ser utilizado em diferentes páginas e tipos de produtos, atribuindo uma flexibilidade no FIEX quando aplicado em páginas diferentes de comércio eletrônico.

Em geral, as tabelas 4.8, 4.9 e 4.10 mostram que obtivemos resultados satisfatórios na extração dos dados, isso mostra que o FIEX consegue extrair dados em diferentes estruturas de lojas de comércio eletrônico. Na Tabela 4.9, é verificado uma

Lojas	Registros	Entidades	Acertos	Erros	Precisão
CasasBahia	21	537	376	161	0.70
Colombo	17	251	199	52	0.79
Extra	20	597	336	261	0.56
FasShop	11	447	336	111	0.75
FNAC	18	543	246	297	0.45
Girafa	11	255	231	24	0.91
Magazine	29	803	750	53	0.93
Mamute	22	474	417	57	0.87
Saraiva	11	290	233	57	0.80
PontoFrio	21	484	408	76	0.84
	181	4681			0.762

Tabela 4.9. [Resultados da avaliação por loja de produtos do tipo TV

Lojas	Registros	Entidades	Acertos	Erros	Precisão
CasasBahia	20	245	198	47	0.81
Colombo	12	90	71	19	0.79
Extra	20	223	190	33	0.85
FasShop	10	119	103	16	0.86
FNAC	8	112	57	55	0.51
Girafa	12	134	120	14	0.89
Magazine	20	283	240	43	0.84
Mamute	12	142	127	15	0.89
Saraiva	10	110	93	17	0.84
PontoFrio	8	94	80	14	0.85
	132	1552			0.81

Tabela 4.10. [Resultados da avaliação por loja de produtos do tipo Smartphones

baixa precisão na extração de dados da loja Extra e FNAC, de modo similar aos problemas descritos na seção anterior, essas lojas apresentavam uma informação adicional nos registros do produtos. Essa informação descreve com um pequeno texto as principais características dos produtos e estavam posicionadas logo após o registro dos produtos. Esse tipo de informação foi considerada como incorreta o que prejudicou o resultado final. Este tipo de informação não estava presente nos tipos de produto de Notebook e Smartphones, por isso, estas duas lojas não tiveram problemas na extração como demonstrado pelos valores de precisão nas Tabelas 4.8 e 4.10.

4.4.4 Avaliação Geral

A extração de dados em páginas Web contendo casos severos de elementos textuais compostos utilizando alinhamento de dados não é possível devido à inconsistência en-

tre a estrura HTML e a representação dos dados da extração, conforme discutido no Capítulo anterior. Nossa abordagem é baseada apenas no conteúdo textual destas páginas Web e pela quantidade de lojas utilizadas na coleção de teste, podemos afirmar que a extração de registros realizada pelo FIEX é flexível, pois funciona em uma grande maioria de distintas lojas de comércio eletrônico. Além disso, os resultados obtidos nas três avaliações feitas, o FIEX consegue inicialmente, extrair registros e posteriormente os dados destes registros de páginas Web de modo eficiente. Nossos testes realizados no domínio de descrição de produtos mostram que é possível extrair esses dados utilizando apenas o conteúdo textual destas páginas.

Nos testes, foi encontrado um problema no qual algumas informações extraídas não pertencem à descrição do produto, ou seja, existe a presença de ruídos que prejudicam o resultado da extração. Estes ruídos foram encontrados após os valores de cada registro em uma mesma página. As informações textuais de tipos de produtos, filtros, e partes do menu presente em algumas lojas, são as maiores fontes de adição de ruído na informação que se deseja extrair. Este problema, é observado quando comparamos as Tabelas 4.8 e 4.10, referentes a produtos de Notebook e Smartphones respectivamente. Os resultados da extração de TV são piores em relação aos de Notebook, isto é ocasionado pela presença de informação irrelevante posicionada logo após aos dados de interesse para a extração. Um exemplo, deste problema aconteceu na extração dos dados da loja FNAC, mesmo problema relatado na extração do registro para as lojas Kabum e Lojas MM, a precisão da extração dos dados é baixa devido a irregularidade na escrita da descrição dos registros de uma mesma página, com isso, a fase de refoço do FIEX não obteve bons resultados, isso deve-se ao fato desta fase utilizar características de posição e sequência dos atributos na qual é criado o modelo PSM pelo ONDUX. As posições desses atributos são muito irregulares, ou seja, não constituem um padrão de estrutura visível, e o método acaba falhando. A solução para este problema, consiste em adicionar à base de conhecimento um atributo qualquer contendo essas informações irrelevantes. A correção deste problema não foi implementada neste trabalho, e será discutida juntamente com as conclusões finais que serão apresentadas no próximo capítulo.

Capítulo 5

Conclusão

Nesta dissertação, apresentamos o método FIEX (**F**ormat **I**ndependent WebData **Ex**traction) que é capaz de extrair registros e seus respectivos dados em casos severos de elementos textuais compostos, cenários que métodos tradicionais falham devido à inconsistência entre estrutura lógica das páginas Web e a estrutura conceitual dos dados nelas representadas. O FIEX utiliza uma abordagem independente de formato para tarefas de extração não-supervisionada de páginas Web e para isso, utiliza-se de técnicas de eliminação de ruído por redundância de informação e técnicas de extração de informação por segmentação de texto (IETS).

Levar em consideração apenas a informação textual de páginas Web para extração de dados demonstrou-se bastante efetiva na prática. O FIEX, trata de maneira conveniente o problema de extração de dados em Páginas Web, alcançando elevados índices de desempenho em todas as coleções testadas.

Em nossos experimentos utilizamos coleções com diversas páginas Web de diferentes sites de comércio eletrônico contendo descrições de produtos em três diferentes domínios de produtos. Avaliamos as duas principais fases do FIEX: a extração de registros e a extração de dados. Na primeira, avaliamos a qualidade da extração dos registros e na segunda, avaliamos a qualidade da extração dos dados em relação à atribuição de rótulos aos termos dos registros. Com os resultados obtidos em nossos experimentos, validamos que a extração em páginas Web ricas em dados sem a utilização de características de estrutura, é possível e eficaz, pois conseguimos resolver problemas das abordagens tradicionais quando não se tem delimitadores para os atributos ou a estrutura das páginas não representa os dados finais de extração, cenário muito comum da Web.

Como trabalhos futuros podemos listar três vertentes. Inicialmente, pretendemos utilizar métodos para detectar automaticamente o início e fim dos registros utilizando

um método conhecido como JUDIE *(Joint Unsupervised Structure Discovery and Information Extraction)* [Cortez et al., 2011]. Em seguida, pretendemos realizar experimentos entre a geração de blocos baseada em NGRAM implementada neste trabalho e a geração de blocos nativa deste método. Por último, pretendemos testar se existe melhora no resultado da extração quando unimos todas as páginas de um mesmo tipo de produto e de um mesmo site antes de aplicar o ONDUX. Assim poderíamos ter mais exemplos, o que pode auxiliar principalmente a etapa de reforço deste método.

Referências Bibliográficas

[Agichtein and Ganti, 2004] Agichtein, E. and Ganti, V. (2004). Mining Reference Tables for Automatic Text Segmentation. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 20–29, Seattle, USA.

[Baeza-Yates and Ribeiro-Neto, 2011] Baeza-Yates, R. and Ribeiro-Neto, B. (2011). *Modern Information Retrieval: The Concepts and Technology Behind Search.* Addison-Wesley Publishing Company, USA, 2nd edition.

[Bizer et al., 2013] Bizer, C., Eckert, K., Meusel, R., Muhleisen, H., Schuhmacher, M., and Volker., J. (2013). Deployment of rdfa, microdata, and microformats on the web: a quantitative analysis. In *The Semantic Web - ISWC 2013*, pages 17–32.

[Blog, 2011] Blog, G. O. (2011). Introducing schema. org: Search engines come together for a richer web r guha. In *The Semantic Web - ISWC 2013*.

[Borkar et al., 2001] Borkar, V., Deshmukh, K., and Sarawagi, S. (2001). Automatic segmentation of text into structured records. In *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, pages 175–186.

[Chang et al., 2006] Chang, C. H., Kayed, M., Girgis, M., and Shaalan, K. (2006). A survey of web information extraction systems. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1411–1428.

[Chaudhuri and Das, 2003] Chaudhuri, S. and Das, G. (2003). Automated ranking of database query results. In *In CIDR*, pages 888–899.

[Cohen and Sarawagi, 2004] Cohen, W. W. and Sarawagi, S. (2004). Exploiting dictionaries in named entity extraction: combining semi-markov extraction processes and data integration methods. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 89–98, Seattle, USA.

[Constantine et al., 2005] Constantine, M., Mehmet, O., and Steve, C. (2005). Separating xhtml content from navigation clutter using dom-structure block analysis. In *Cora Information Extraction Collection*.

[Cortez et al., 2010a] Cortez, E., da Silva, A., Gonçalves, M., and de Moura, E. (2010a). Ondux: On-demand unsupervised learning for information extraction. In *SIGMOD 2010*, pages 807–818, Indianapolis, USA.

[Cortez et al., 2011] Cortez, E., da Silva, A. S., , de Moura, E. S., and Laender, A. H. F. (2011). Joint unsupervised structure discovery and information extraction. In *Proceedings of the ACM SIGMOD International Conference on Management of Data Conference*, pages 541–552, Athens, Greece.

[Cortez and da Silva, 2013] Cortez, E. and da Silva, A. S. (2013). *Unsupervised Information Extraction by Text Segmentation*. Springer Briefs in Computer Science. Springer.

[Cortez et al., 2010b] Cortez, E., da Silva an Marcos AndrÃ© GonÃ§alves, A. S., and de Moura, E. S. (2010b). Ondux: On-demand unsupervised learning for information extraction. In *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, pages 807–818.

[Dalvi et al., 2009] Dalvi, N., Bohannon, P., and Sha, F. (2009). Robust web extraction: An approach based on a probabilistic tree-edit model. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pages 335–348, New York, NY, USA. ACM.

[Ferrara et al., 2014] Ferrara, E., Meo, P. D., Fiumara, G., and Baumgartner, R. (2014). Web data extraction, applications and techniques: A survey. *Knowledge-Based Systems*, 70:301–323.

[Gulhane et al., 2010] Gulhane, P., Rastogi, R., Sengamedu, S. H., and Tengli, A. (2010). Exploiting content redundancy for web information extraction. In *Proc. Very Large Database Endowment*, pages 578–587.

[jsoup Java HTML Parser,] jsoup Java HTML Parser. http://jsoup.org/.

[Laender et al., 2002] Laender, A. H. F., Ribeiro-Neto, B. A., da Silva, A. S., and Teixeira, J. S. (2002). A brief survey of web data extraction tools. *SIGMOD Record*, 31:84–93.

[Lafferty et al., 2001a] Lafferty, J., McCallum, A., and Pereira, F. C. (2001a). Probabilistic models for segmenting and labeling sequence data. In *In Proc. of the Eighteenth International Conference on Machine Learning*, pages 282–289.

[Lafferty et al., 2001b] Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001b). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Liu et al., 2010] Liu, W., Meng, X., and Meng, W. (2010). Vide: A vision-based approach for deep web data extraction. *Knowledge and Data Engineering, IEEE Transactions on*, 22(3):447–460.

[Lu et al., 2007a] Lu, Y., He, H., Zhao, H., Meng, W., and Yu, C. (2007a). Annotating structured data of the deep web. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 376–385.

[Lu et al., 2007b] Lu, Y., He, H., Zhao, H., Meng, W., and Yu, C. (2007b). Annotating structured data of the deep web. In *IEEE 23rd International Conference on Data Engineering, 2007. ICDE 2007.*, pages 376–385.

[Lu et al., 2013] Lu, Y., He, H., Zhao, H., Meng, W., and Yu, C. (2013). Annotating search results from web databases. *Knowledge and Data Engineering, IEEE Transactions on*, 25(3):514–527.

[Mesquita et al., 2007] Mesquita, F., da Silva, A. S., de Moura, E. S., Calado, P., and Laender, A. H. F. (2007). Labrador: Efficiently publishing relational databases on the web by using keyword-based query interfaces. *Inf. Process. Manage.*, 43(4):983–1004.

[Miao et al., 2009] Miao, G., Tatemura, J., Hsiung, W.-P., Sawires, A., and Moser, L. E. (2009). Extracting data records from the web using tag path clustering. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 981–990.

[Neemu.com,] Neemu.com. Empresa de tecnologia inteligente para e-commerce.

[Parser, a] Parser, H. http://htmlparser.sourceforge.net/.

[Parser, b] Parser, J. J. http://jericho.htmlparser.net.

[Pedralho and da Silva, 2001] Pedralho, A. S. and da Silva, A. S. (2001). ExtraÃ§Ã£o automÃ¡tica de dados de pÃ¡ginas html utilizando alinhamento em dois nÃveis. Mestrado, Universidade Federal do Amazonas, Departamento de CiÃªncia da ComputaÃ§Ã£o, UFAM.

[Pereira and Silva, 2006] Pereira, D. O. and Silva, A. S. (2006). GeraÃ§Ã£o semi-automÃ¡tica de extratores de dados web considerando contextos fracos. Master's thesis, Universidade Federal do Amazonas, Instituto de CiÃªncias Exatas, Departamento de CiÃªncia da ComputaÃ§Ã£o.

[Peters and Lecocq, 2013] Peters, M. and Lecocq, D. (2013). Content extraction using diverse feature sets. In *Proceedings of the 22Nd International Conference on World Wide Web Companion*, pages 89–90.

[Raghavan and Garcia-Molina, 2001] Raghavan, S. and Garcia-Molina, H. (2001). Crawling the hidden web. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 129–138, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Reis et al., 2004] Reis, D. C., Golgher, P. B., Silva, A. S., and Laender, A. F. (2004). Automatic web news extraction using tree edit distance. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 502–511.

[Sarawagi, 2008] Sarawagi, S. (2008). Information extraction. *Foundations and Trends in Databases*, 1(3):261–377.

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

[Valiente, 2002] Valiente, G. (2002). Tree edit distance and common subtrees. *Research Report LSI-02-20-R.*

[Wang and Lochovsky, 2003] Wang, J. and Lochovsky, F. H. (2003). Data extraction and label assignment for web databases. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, pages 187–196, New York, NY, USA. ACM.

[Weninger et al., 2012] Weninger, T., Hsu, W. H., and Han, J. (2012). Content extraction with tag ratios. In *Proceedings of the 19th International Conference on World Wide Web*, pages 971–980.

[Yang, 1991] Yang, W. (1991). Identifying syntactic differences between two programs. *Softw. Pract. Exper.*, 21(7):739–755.

[Zhai and Liu, 2005] Zhai, Y. and Liu, B. (2005). Web data extraction based on partial tree alignment. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 76–85, New York, NY, USA. ACM.

[Zhao et al., 2005] Zhao, H., Meng, W., Wu, Z., Raghavan, V., and Yu, C. (2005). Fully automatic wrapper generation for search engines. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 66–75, New York, NY, USA. ACM.