UNIVERSIDADE FEDERAL DE SANTA CATARINA
INSTITUTO DE INFORMÁTICA E ESTATÍSTICA

Larissa Rodrigues Lautert

WT2RT: TAXONOMIA, CLASSIFICAÇÃO E ALGORITMOS PARA UNIFORMIZAÇÃO DE TABELAS WEB

Florianópolis (SC)

2013

Larissa Rodrigues Lautert

WT2RT: TAXONOMIA, CLASSIFICAÇÃO E ALGORITMOS PARA UNIFORMIZAÇÃO DE TABELAS WEB

Dissertação submetido ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do Grau de mestre.
Orientadora: Carina Friedrich Dorneles, Dra.

Florianópolis (SC)

2013

Catalogação na fonte elaborada pela biblioteca da
Universidade Federal de Santa Catarina

A ficha catalográfica é confeccionada pela Biblioteca Central.

Tamanho: 7cm x 12 cm

Fonte: Times New Roman 9,5

Maiores informações em:
http://www.bu.ufsc.br/design/Catalogacao.html

RESUMO

A Web é o maior repositório de dados disponível, contando com mais de 150 milhões de tabelas com dados relacionais de qualidade. Muitos trabalhos têm unido esforços a fim de utilizá-las como base para consultas, porém, a heterogeneidade de formatações em que os dados se encontram limita a 17,75% a quantidade de tabelas aptas para este tipo de processamento. A fim de aumentar o aproveitamento das informações estruturadas na Web, esta dissertação apresenta o WT2RT (*Web Table to Relational Table*), uma solução para catalogação das categorias de tabelas utilizadas com maior frequência, formalização destas e definição de algoritmos para uniformização estrutural. Para a catalogação, foi implementado o *framework* WTCLASSIFIER, baseado em Redes Neurais Artificiais. Seu aprendizado se dá através da análise de padrões em tabelas, escolhidas aleatoriamente, cujas categorias são conhecidas. Nos experimentos realizados, o WTCLASSIFIER apresentou valores altos de *F-measure* para a maioria das estruturas definidas. Após a categorização, são aplicados algoritmos em cada caso heterogêneo, de modo a trazer todas as tabelas para uma estrutura única.

Palavras-chave: tabelas Web, tabelas heterogêneas, taxonomia de estrutura, classificação automática, algoritmos de uniformização.

ABSTRACT

The Web is the largest repository of available data, with over 150 million high-quality tables. Several works have combined efforts to allow queries on these tables, however, their heterogeneous structures limit to 17.75% the amount of tables suitable for this type of processing. In order to increase the use of structured information on the Web, this work presents an approach called WT2RT (Web Table to Relational Table), that catalogs Web table categories used more often, formalize them and defines structural uniformization algorithms. For the cataloging purpose, framework WTCLASSIFIER was implemented, using Artificial Neural Networks. It learns analyzing patterns of tables whose categories are known. In experiments, WTCLASSIFIER presented high F-measure values for most cases. After identifying table category, algorithms are applied in order to bring all tables to a single structure.

Keywords: Web tables, heterogeneous tables, structural taxonomy, automatic classification, uniformization algorithms.

LISTA DE FIGURAS

Figura 1 Tabela Web com formatação diferente da Relacional. 13
Figura 2 Um neurônio abstrato. 15
Figura 3 Multilayer Perceptron. 16
Figura 4 Taxonomia proposta por Yoshida et al. 19
Figura 5 Taxonomia proposta por Crestan et al. 20
Figura 6 Junção *merge*. 26
Figura 7 Algoritmo para extração da tabela HTML. 28
Figura 8 Algoritmo de transposição de tabela. 28
Figura 9 Algoritmo que obtém determinada coluna como linha. 28
Figura 10 Algoritmo que transpõe a tabela durante sua extração. 30
Figura 11 Gráficos com o tempo de execução em função do número de linhas das tabelas utilizadas para as três abordagens. 31
Figura 12 Classificação hierárquica dos tipos de formatação de Web tables. 34
Figura 13 Exemplos de Web tables 36
Figura 14 Exemplo de Matrix Web table. 37
Figura 15 Exemplo de Concise Web table onde as *merged cells* atuam como *sub-headers*. 38
Figura 16 Exemplo de Concise Web table onde as *merged cells* evitam repetição de valores. 38
Figura 17 Exemplo de Nested Web table. 39
Figura 18 Exemplo de Splitted Web table. 40
Figura 19 Exemplo de Navigational e Formatting Web table. 41
Figura 20 Vertical Web table representada como Tabela Relacional. 44
Figura 21 Matrix Web table representada como Tabela Relacional. 47
Figura 22 Concise Web table representada como Tabela Relacional. 47
Figura 23 Concise Web table representada como Tabela Relacional. 47
Figura 24 Nested Web table representada como duas Tabelas Relacionais. 48
Figura 25 Splitted Web table representada como Tabela Relacional. 50
Figura 26 Gráfico de revocação vs precisão para a Classificação Principal. 57
Figura 27 Gráfico de revocação vs precisão para a Classificação Se-

cundária. 58
Figura 28 Distribuição de Web tables na Classificação Primária. 59
Figura 29 Distribuição de Web tables na Classificação Secundária. 59

LISTA DE TABELAS

Tabela 1 Estrutura de uma Web table. 34
Tabela 2 Comparação de categorias de tabelas mencionadas em trabalhos. 51
Tabela 3 Comparação de categorias de tabelas mencionadas em trabalhos. 53
Tabela 4 Comparação dos resultados do WTClassifier com os esperados para a Classificação Principal. 56
Tabela 5 Comparação dos resultados do WTClassifier com os esperados para a Classificação Secundária. 56
Tabela 6 Precisão, revocação e medida F para a Classificação Principal. 57
Tabela 7 Precisão, revocação e medida F para a Classificação Secundária. 57
Tabela 8 Medida F do TabEx e do WTClassifier. 58

SUMÁRIO

1 INTRODUÇÃO 13
2 TRABALHOS RELACIONADOS 15
2.1 REDES NEURAIS ARTIFICIAIS 15
2.2 TABELAS NA WEB 17
2.3 CATEGORIZAÇÃO DE TABELAS WEB 18
2.3.1 Integração de tabelas Web 18
2.3.2 TabEx 19
3 PRÉ-PROCESSAMENTO DE TABELAS WEB 25
3.1 ALGORITMO DE JUNÇÃO *MERGE* 25
3.2 ESTRATÉGIAS PARA EXECUÇÃO DO ALGORITMO DE JUNÇÃO MERGE ENTRE TABELAS WEB HETEROGÊNEAS 26
3.2.1 Abordagem 1: Transposição de tabela após extração 27
3.2.2 Abordagem 2: Transposição de tabela durante extração 27
3.2.3 Abordagem 3: Adaptação do algoritmo de junção *merge* 28
3.3 EXPERIMENTOS 29
4 WT2RT: TAXONOMIA, FORMALIZAÇÃO E ALGORITMOS PARA UNIFICAÇÃO DE WEBTABLES 33
4.1 CLASSIFICAÇÃO DE TABELAS WEB 33
4.1.1 Conceitos Gerais 33
4.1.2 Relational Knowledge Web Tables 35
4.1.3 Layout Web Tables 40
4.2 WTCLASSIFIER, UM CLASSIFICADOR BASEADO EM REDES NEURAIS ARTIFICIAIS 40
4.2.1 Funcionamento básico do WTCLASSIFIER 41
4.2.2 Implementação 42
4.3 ALGORITMOS PARA UNIFICAÇÃO ESTRUTURAL 43
4.3.1 Vertical Web Table 43
4.3.2 Matrix Web Table 44
4.3.3 Concise Web Table 45
4.3.4 Nested Web Table 47
4.3.5 Splitted Web Table 48
4.3.6 Simple Multivalued Web Table 50
4.3.7 Composed Multivalued Web Table 50
4.4 COMPARAÇÃO COM TRABALHOS RELACIONADOS A TABELAS WEB 50
5 ANÁLISE EXPERIMENTAL 55
5.1 AVALIAÇÃO QUALITATIVA DO WTCLASSIFIER 55

5.2 DISTRIBUIÇÃO DE TABELAS WEB DE ACORDO COM A TAXONOMIA PROPOSTA 58
6 CONCLUSÕES E TRABALHOS FUTUROS 61
Referências Bibliográficas 63

1 INTRODUÇÃO

De acordo com Cafarella et al. (CAFARELLA et al., 2008), há mais de 150 milhões de tabelas HTML na Web com dados relacionais de qualidade. O maior incentivo para trabalhar com essas informações é permitir análise e integração de dados na Web, já que esta é a maior base de tabelas existente. Muitos trabalhos têm unido esforços para identificar (WANG; HU, 2002a, 2002b; EMBLEY; TAO; LIDDLE, 2005), extrair (GATTERBAUER et al., 2007; CAFARELLA; MADHAVAN; HALEVY, 2009; NAGY et al., 2011) e rotular (SILVA et al., 2007; VENETIS et al., 2011) esses dados, de modo que possam ser usados para base de consultas estruturadas. Entretanto, muitas destas tabelas têm formatação diferente das tabelas de Bancos de Dados Relacionais, o que impede a aplicação direta de consultas estruturadas. Por isso, tabelas Web precisam ser analisadas individualmente para ter suas estruturas compreendidas.

A Figura 1 mostra uma estrutura com dados relacionais de qualidade encontrada na Wikipedia. Analisando sua formatação, percebem-se características que a diferenciam de uma tabela relacional. Primeiro, os nomes dos atributos estão dispostos verticalmente, e não em uma linha, como no Modelo Relacional. Além disso, há duas tabelas aninhadas (*General information* e *Broadcast*) e os atributos *Developed by* e *Language(s)* são multivalorados. Neste caso, os dados são facilmente compreendidos por humanos, mas não por computadores.

Devido aos diversos tipos de estruturas encontrados na Web, é ne-

General information	
Title	Game of Thrones
Genre	Medieval fantasy
Developed by	David Benioff D. B. Weiss
Language(s)	English, Dothraki
Broadcast	
Original channel	HBO
Picture format	1080i (HDTV)
Audio format	Dolby Digital 5.1

Figura 1: Tabela Web com formatação diferente da Relacional.

cessário catalogar e interpretar as categorias mais utilizadas. Assim, elas terão seu entendimento e processamento simplificados. Com esse propósito, Crestan et al. (CRESTAN; PANTEL, 2011) propôs uma taxonomia com nove estruturas heterogêneas, porém, alguns tipos não foram incluídos. Dessa forma, a presente dissertação propõe uma extensão ao trabalho de Crestan, com uma taxonomia mais completa e formalizações para cada categoria de tabela Web com dados relacionais. Apresenta-se o WTCLASSIFIER, um classificador baseado em Redes Neurais Artificiais, que aprende através da análise das características de cada categoria.

Esta dissertação possui as seguintes contribuições: (i) definição de uma taxonomia para tabelas Web heterogêneas; (ii) formalizações de suas categorias com dados relacionais; (iii) *framework* para classificação de tabelas Web heterogêneas; e (iv) algoritmos para unificação destas estruturas heterogêneas.

O restante deste trabalho está organizado como segue. O Capítulo 2 traz uma análise dos trabalhos relacionados sobre tabelas Web. Experimentos e motivações preliminares, referentes a pré-processamento de tabelas heterogêneas, são apresentados no Capítulo 3. A solução proposta nesta dissertação, intitulada WT2RT, está no Capítulo 4, onde três seções detalham a taxonomia proposta para tabelas Web, detalhes do *framework* WTCLASSIFIER e algoritmos para unificação estrutural. A avaliação experimental é mostrada no Capítulo 5, junto com a classificação de 342.795 tabelas coletadas da Web. Finalmente, o Capítulo 6 apresenta conclusões e trabalhos futuros.

2 TRABALHOS RELACIONADOS

Este capítulo apresenta, primeiramente, uma fundamentação teórica sobre Redes Neurais Articiais (Seção 2.1), utilizadas na implementação deste trabalho. Após, traz uma revisão bibliográfica dos principais trabalhos relacionados a dados semi-estruturados na Web (Seção 2.2) e categorização de tabelas Web (Seção 2.3).

2.1 REDES NEURAIS ARTIFICIAIS

Inspirados no sistema nervoso humano, McCulloch and Pitts (MCCULLOCH; PITTS, 1943) criaram Redes Neurais Artificiais (RNA): modelos computacionais capazes de processar informação, aprender e reconhecer padrões. Esses modelos são compostos por elementos de processamento simples, chamados de *neurônios*. A Figura 2 mostra a estrutura de um neurônio abstrato com n entradas e uma função de ativação f, computada internamente. Seus canais de entrada possuem pesos associados, cujos valores são multiplicados pelas respectivas entradas. A informação é integrada no neurônio, usualmente somando os n sinais, e a função f é calculada (ROJAS, 1996).

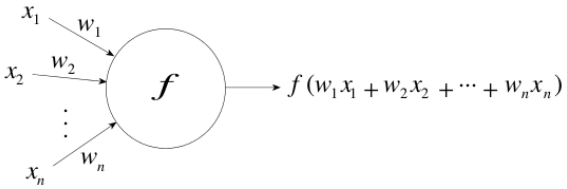


Figura 2: Um neurônio abstrato.

Em RNAs, cada neurônio possui uma função primitiva, normalmente sigmoidal ou tangente hiperbólica, capaz de transformar sua(s) entrada(s) em saída(s). Os neurônios encontram-se altamente interconectados, formando camadas. Cada camada representa uma combinação não-linear de funções não-lineares da camada anterior. A primeira interage com o ambiente para receber as entradas, enquanto a última apresenta os dados processados. Em cada transmissão de dados entre neurônios, os valores são multiplicados pelos pesos das conexões. Podem haver camadas entre essas duas, que não interagem com o ambiente externo. Elas são chamadas de camadas escondidas, e aumentam a capacidade computacional da rede. Um exemplo dessa arquitetura, conhecida como *Multilayer Perceptron*, é exibida na Figura 3.

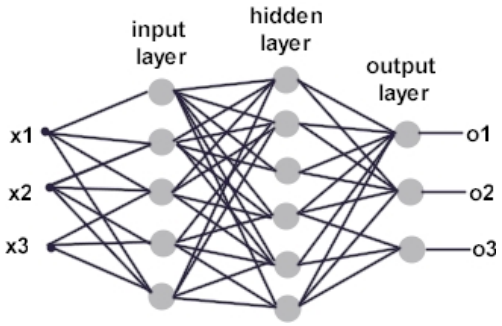


Figura 3: Multilayer Perceptron.

RNAs podem ser utilizadas para aproximar funções, reconhecer padrões, agrupar, controlar processos industriais, analisar e processar sinais. Sua principal característica é a de aprender, ou seja, utilizar um conjunto de observações para inferir o mapeamento implícito dos dados. Considere, por exemplo, que se conhece uma função em apenas alguns pontos e deseja-se generalizar para outros. O processo de aprendizagem consiste em adaptar os parâmetros da rede da melhor forma para refletir a informação conhecida e, após, extrapolar para as entradas desconhecidas desejadas.

Um algoritmo de aprendizagem pode ser supervisionado ou não-supervisionado. O primeiro consiste em um método onde tem-se exemplos de entrada das quais se conhece a saída esperada. Estas são apresentadas para a rede, que utiliza o erro (diferença entre saídas calculada e esperada), junto com a entrada, para ajustar os pesos dos canais de entrada. Esse processo é repetido até que seja encontrada a melhor solução, ou seja, a combinação dos pesos que minimiza a função do erro. Uma função comumente utilizada para minimizar a diferença entre a saída da rede e a saída esperada é o *erro médio quadrático*. Um dos algoritmos mais conhecidos de aprendizagem é o de *Backpropagation*, que minimiza o erro através do método de *gradient descent*. Uma otimização deste algoritmo é o *Resilient Backpropagation*, que leva em conta apenas o sinal do erro, e não sua magnitude, para atualizar os valores dos pesos no processo de aprendizagem (RIEDMILLER; BRAUN, 1993). Já na aprendizagem não-supervisionada, a saída desejada é desconhecida e tem-se o desafio de encontrar estruturas em dados não rotulados, baseado nas características dadas.

Neste trabalho, foi utilizada uma rede neural com uma camada de entrada, uma escondida e uma de saída (*Multilayer Perceptron*). Na primeira, cada neurônio corresponde a uma das característica utilizadas como entrada.

Como seu propósito era de classificação, cada neurônio na camada de saída representa uma categoria. O algoritmo de aprendizagem aplicado foi *resilient backpropagation*, por sua característica de adaptabilidade a diferentes propósitos.

2.2 TABELAS NA WEB

Trabalhos que acessam conteúdo de tabelas Web vem sendo desenvolvidos há algum tempo, dada a vasta quantidade de dados relacionais presentes nelas (SILVA et al., 2007; CAFARELLA et al., 2008; CAFARELLA; MADHAVAN; HALEVY, 2009; MERGEN; FREIRE; HEUSER, 2008; LIMAYE; SARAWAGI; CHAKRABARTI, 2010; MERGEN; FREIRE; HEUSER, 2010; VENETIS et al., 2011).

Em 2008, Cafarella et al. (CAFARELLA et al., 2008) extraíram 14.1 bilhões de tabelas HTML com o *Web crawler* do Google e, após aplicarem algumas técnicas estatísticas, estimaram que 154 milhões destas possuíam dados relacionais de alta qualidade. Este número é 5 ordens de magnitude maior do que qualquer banco de dados conhecido. O objetivo do trabalho foi buscar documentos com dados estruturados de qualidade e analisar quais as vantagens decorrentes da manipulação de tal volume de tabelas. Com isso, foi desenvolvida uma técnica para ordenar um *ranking* de tabelas em resposta a uma consulta por palavra-chave, a qual foi de 85 a 98% melhor do que usar motores de busca tradicionais para este propósito. Através de estatísticas referentes a correlações de atributos, outra contribuição foi uma ferramenta para auxiliar na criação e exploração de esquemas de bancos de dados. Quanto ao aspecto de criação, imagine que um *designer* de banco de dados está modelando uma tabela e digita "Título" e "Editora". A ferramenta irá sugerir "Autor", pois esses termos costumam aparecer juntos. Entretanto, se "Título" aparecesse junto com "Diretor", provavelmente se trataria de filmes. A ferramenta ainda permite que um usuário, ao acessar essa tabela, possa navegar para outra com dados de Editoras, através de um *link* junção gerado automaticamente.

Para a realização de consultas sobre tabelas HTML, o protótipo *Structured Web Search Engine* (MERGEN; FREIRE; HEUSER, 2008) foi projetado tendo em vista a escala e a natureza dinâmica das fontes de dados Web. Por isso, os mapeamentos entre as fontes de dados são derivados automaticamente através do casamento entre os atributos da consulta e das tabelas. Esses termos encontram-se armazenados em um índice invertido bipartido. As consultas devem ser expressas em linguagem SQL, com suporte às operações de *seleção*, *projeção* e *junção*. A limitação desta abordagem consiste nas ta-

belas utilizadas como base de dados: são consideradas apenas formatações tradicionais.

O maior problema destas estruturas de dados vem do fato de elas serem feitas para interpretação humana, e por isso, computadores têm dificuldade em entender alguns tipos de formatação. Estruturas heterogêneas de tabelas HTML, como tabelas com tuplas transpostas e células mescladas, já foram citadas em trabalhos anteriores (CAFARELLA; WU, 2008; VENETIS et al., 2011), porém, a maioria deles apenas se preocupou em detectar tabelas com formatação semelhante à relacional, (CAFARELLA et al., 2008; CAFARELLA; WU, 2008; WANG; HU, 2002b). Entretanto, dois trabalhos na literatura propõe-se a classificar tabelas com estruturas heterogêneas. Estes são detalhados na seção a seguir.

2.3 CATEGORIZAÇÃO DE TABELAS WEB

A fim de extrair conhecimento contido nas tabelas Web, é necessário catalogar os tipos mais utilizados e entender a forma como eles organizam seus dados. Essa atividade foi desenvolvida por dois trabalhos na literatura, descritos a seguir.

2.3.1 Integração de tabelas Web

Yoshida et al. (YOSHIDA; TORISAWA, 2001) apresentaram um método para integrar tabelas Web que possuem conteúdo relacional e propuseram separá-las em nove categorias. Seu trabalho apresenta um método para unir informações sobre temas similares distribuídos em documentos Web. O objetivo final é, a partir de tabelas heterogêneas sobre objetos similares, construir uma tabela única com todas essas informações. Para isso, é preciso reconhecer a estrutura de cada tabela e agrupar as que se referem a objetos similares. Na primeira tarefa, assumem que o conteúdo de uma célula pode ser um atributo (*label*) ou valor (*data*), e baseiam a classificação estrutural de acordo com essas posições. Através do algoritmo de Maximização de Expectativa (DEMPSTER; LAIRD; RUBIN, 1977), é estimada a probabilidade de cada termo aparecer como atributo ou valor e então a estrutura da tabela é determinada de acordo com essas probabilidades. Após, agrupam tabelas que tratam de objetos similares e, por isso, podem ser integradas. As categorias propostas são ilustradas na Figura 4.

As categorias da Figura 4 foram propostas com base na posição dos atributos em cada tabela. No tipo 0 a tabela não possui atributos, enquanto que

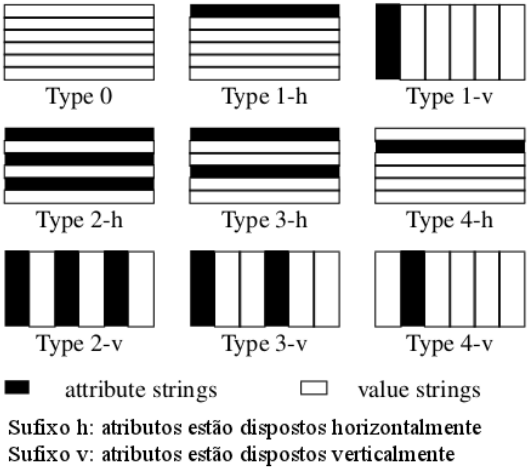


Figura 4: Taxonomia proposta por Yoshida et al.

nos tipos 1-h e 1-v, apenas a primeira linha/coluna, respectivamente, possui atributos. Para os outros casos, o número junto ao nome do tipo indica a cada quantas linhas encontra-se uma linha/coluna de atributos, representadas em preto na Figura 4. Por exemplo, no tipo 2-h, a cada duas linhas, a linha é composta por atributos. Já em 3-h, isso ocorre a cada três linhas. A mesma lógica repete-se para os tipos 2-v, 3-v, 4-h e 4-v.

A classificação de Yoshida é interessante por considerar tabelas heterogêneas e citar estruturas transpostas, porém, como ponto negativo, destaca-se o número associado ao nome de cada categoria. Se fosse encontrada uma tabela em que os atributos repetissem-se a cada 5 linhas, por exemplo, não se enquadraria na taxonomia proposta. Como há uma enorme heterogeneidade de estruturas na Web, não há garantia de que casos como esse não ocorram.

2.3.2 TabEx

Crestan et al. (CRESTAN; PANTEL, 2011) propõem outra taxonomia e, de acordo com ela, relatam um censo de tabelas HTML presentes na Web. O foco do trabalho são Tabelas Relacionais compostas por triplas semânticas da forma $<p, s, o>$, onde p é um *predicado*, ou uma relação, s é o *sujeito* do predicado e o é seu *objeto*. Por exemplo, na Figura 1, a segunda linha seria representada como $<$*General information*, *Title*, *Game of Thrones*$>$.

Através de um *Web crawler*, 1,2 bilhões de páginas foram visitadas, sendo que 896 milhões continham tabelas HTML. Destas, foram retiradas 12 bilhões de tabelas. Com o intuito de minimizar estruturas de leiaute, foram descartadas as que continham outras tabelas em seu interior. Após esse processo, restaram 8,2 bilhões, das quais 2,6 bilhões eram únicas.

Com base na análise destas tabelas, propuseram a taxonomia da Figura 5, mais completa e atualizada do que a de Yoshida. As categorias cobrem desde propósito relacional à leiaute. Abaixo, são descritas as categorias propostas.

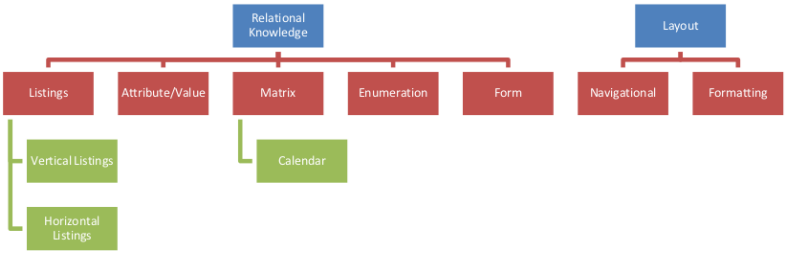


Figura 5: Taxonomia proposta por Crestan et al.

Categorias **Relational Knowledge**:

- **Vertical Listing**: listam um ou mais atributos de entidades similares em tuplas dispostas horizontalmente. Exemplo: listagem de capital e população (predicados) de países (sujeito). Cada valor de capital e população é um objeto.
- **Horizontal Listing**: listam um ou mais atributos de entidades similares, porém as tuplas são dispostas verticalmente. Exemplo: comparação de características (predicados) de câmeras fotográficas (sujeito).
- **Attribute/Value**: caso específico de *Vertical/Horizontal Listing* que possui apenas uma tupla. Frequentemente não contém o sujeito na tabela, e sim no contexto da página que a contém. Exemplo: tabela com especificações (predicados) de um computador (sujeito).
- **Matrix**: apresenta o mesmo tipo de dado em cada célula na junção de uma coluna com uma linha. Normalmente os cabeçalhos de linha e coluna contêm os sujeito, mas o objeto somente é obtido combinando os dois sujeitos. Com frequência, o predicado não está contido explicitamente na tabela. Exemplo: número de acidentes de trânsito (predicado) por mês, nas linhas, e por estado, nas colunas (predicados).

- **Calendar**: caso específico de matriz, diferenciando apenas na semântica. O sujeito é a data e os predicados geralmente são relações do tipo "ocorre em". Exemplo: calendário listando shows em um teatro (sujeito) de acordo com suas datas (predicados). Nesse caso, existe o desafio de extrair o sujeito, que poderia estar implícito.

- **Enumeration**: caso específico de *Vertical/Horizontal Listing* com apenas um atributo, ou seja, apenas lista objetos que tem a mesma relação ontológica. Exemplo: enumeração dos estados do Brasil (sujeito). Os nomes dos estados constituem os predicados.

- **Form**: similar a Attribute/Value, porém com campos para o usuário preencher. Exemplo: formulário de usuário e senha (predicados) para entrar em um site.

Categorias de **Layout** (não contêm conhecimento relacional):

- **Navigational**: Tabelas compostas por células que organizam um menu de navegação. Exemplo: *links* para categorias de produtos em um site de *e-commerce*.

- **Formatting**: Usada para dispor elementos visualmente na página. Exemplo: alinhar texto e imagens em um site.

Após apresentarem a taxonomia, eles estimaram a proporção de cada categoria na Web. Para isso, automatizaram a classificação, através do *TabEx*, um modelo baseado em árvore de decisão que tem como entrada as características abaixo:

1. Número máximo de linhas em cada coluna.
2. Número máximo de colunas em cada linha.
3. Número máximo de caracteres das células.
4. Comprimento médio das células ao longo de uma coluna/linha.
5. Variância no comprimento da célula em uma coluna/linha.
6. Fração de células em uma coluna/linha geradas por *colspan*.
7. Fração de células em uma coluna/linha geradas por *rowspan*.
8. Fração de *tags* em uma coluna/linha.
9. Fração de células contendo a *tag* de cabeçalho $<th>$ (tabelas com cabeçalho geralmente contêm dados estruturados).
10. Fração de células contendo *links* (tendem a ser *navigational*).

11. Fração de células contendo *tag* de imagem <*img*> (auxilia a identificar tabelas *formatting*).
12. Fração de células contendo *tag* <*input*> (tendem a ser tabelas *form*).
13. Fração de células contendo *tag* <*select*> (auxilia a identificar tabelas *form*).
14. Fração de células contendo alteração na fonte através das *tags* <*b*>, <*u*>, <*font*> e <*i*>.
15. Fração de células contendo *tag* de quebra de linha <*br*>.
16. Fração *strings* distintas em uma coluna/linha.
17. Fração de células terminadas em vírgula (auxilia na identificação de tabelas *attribute/value*).
18. Fração de células contendo números.
19. Fração de células onde o conteúdo é estritamente numérico (costumam indicar *vertical/horizontal listing*).
20. Fração de células não vazias.

Das 8,2 bilhões de tabelas coletadas, 5.000 foram escolhidas e manualmente classificadas de acordo com a taxonomia. Elas foram usadas para o classificador analisar os padrões de entrada de cada tabela e, assim, aprender as características mais significativas de cada categoria. Os resultados são comparados aos desta dissertação na Seção 5.1.

Pode-se notar que, pensando em um algoritmo para trazer tabelas Web heterogêneas para uma estrutura única, as categorias *Attribute/Value* (tabela Web com apenas uma entidade) e *Enumeration* (tabela Web com apenas um atributo) podem ser simplesmente classificadas em *Vertical* ou *Horizontal Listing*. Além disso, *Form* seria melhor classificada fora de *Relational Knowledge Table*, já que, em geral, não possui dados relacionais de qualidade. Antes do processo de classificação, são descartadas tabelas que possuem outras tabelas em seu interior, a fim de minimizar o número de *formatting*. Nesta dissertação, a responsabilidade de decidir quão importante essa característica é para cada categoria foi repassado a uma rede neural utilizada para classificação. Outra questão é que, como tratam-se de tabelas, que são, formalmente, relações (CODD, 1970), seria apropriado defini-las de acordo com a Teoria dos Conjuntos.

A relevância do trabalho de Crestan et al. baseia-se principalmente no volume de tabelas consideradas. Além disso, apresenta uma taxonomia significativamente mais completa e atualizada do que a anterior, de Yoshida. Sua

taxonomia, classificador com *machine learning* e características para aprendizagem inspiraram a realização do presente trabalho, que propõe uma série de melhorias a partir dele.

3 PRÉ-PROCESSAMENTO DE TABELAS WEB

Para encontrar páginas Web em resposta a consultas por palavra-chave, motores de busca tradicionais utilizam a proximidade entre os termos de entrada nas páginas como um dos parâmetros para ordenar os resultados. Dessa forma, não tiram proveito de estruturas em forma de tabelas, onde um termo na última linha pode estar fortemente relacionado a outro na primeira linha, se esta for composta pelos nomes dos atributos. Utilizando tabelas como base de dados, é possível realizar consultas estruturadas no estilo SQL e, assim, aumentar a precisão dos resultados. Trabalhos anteriores já focaram na realização de consultas estruturadas a dados Web com suporte a alguns operadores relacionais, como junção (MERGEN; FREIRE; HEUSER, 2008). Entretanto, nenhum deles utilizou, como base de dados, tabelas com formatação diferente da relacional.

Com o objetivo de maximizar o número de estruturas utilizadas nesse processo, o propósito inicial dessa dissertação era desenvolver uma ferramenta para consultas estruturadas na Web utilizando tabelas com formatações heterogêneas como base de dados. Tal ferramenta teria suporte ao operador de junção, fundamental para estabelecer ligações entre tabelas Web de diferentes páginas e retornar resultados mais completos. Esse capítulo relata experimentos preliminares realizados com um tipo específico de tabela heterogênea. A Seção 3.1 apresenta o algoritmo de junção *merge*, utilizado na Seção 3.2, onde são detalhadas abordagens implementadas de pré-processamento em tabelas Web. Os experimentos referentes a essas implementações são discutidos na Seção 3.3.

3.1 ALGORITMO DE JUNÇÃO *MERGE*

O algoritmo original de junção *merge* pode ser usado para a implementação de junção natural e equi-junção. A seguir, ele é descrito para as relações $r(R)$ e $s(S)$, onde $R \cap S$ representa seus atributos em comum. Por questões de desempenho, o algoritmo supõe que ambas as relações encontram-se ordenadas de acordo com os atributos de $R \cap S$. O algoritmo de junção *merge* mostrado na Figura 6 foi retirado de (SILBERSCHATZ; KORTH; SUDARSHAN, 2006). Nele, *AtribJunção* (linha 12) refere-se aos atributos em $R \cap S$, e $t_r \bowtie t_s$, em que t_r e t_s são tuplas que possuem os mesmos valores para *AtribJunção*, denota a concatenação dos atributos das tuplas. O algoritmo começa associando um ponteiro à primeira tupla de cada relação. Após, armazena as tuplas de s com o mesmo valor para *AtribJunção* e lê

as correspondentes na relação *r*. É importante que as tuplas de *s* caibam na memória principal para evitar transferência de blocos, pois tornaria o processo mais oneroso. Como as duas relações estão ordenadas e o algoritmo as percorre sequencialmente, cada tupla é acessada uma única vez. Assim, o número de acesso a blocos é igual à soma do número de blocos das duas relações.

```
 1 pr := endereço da primeira tupla de r;
 2 ps := endereço da primeira tupla de s;
 3 while (ps ≠ nulo and pr ≠ nulo) do
 4   begin
 5       t_s := tupla para qual ps aponta;
 6       S_s := {t_s};
 7       configure ps para apontar para a próxima tupla de s;
 8       acabou := falso;
 9       while (not acabou and ps ≠ nulo) do
10            begin
11                 t_s' := tupla para qual ps aponta;
12                 if (t_s'[AtribJunção] = t_s[AtribJunção])
13                      then begin
14                           S_s := S_s ∪ {t_s'};
15                           configure ps para apontar para a próxima tupla de s;
16                           end
17                      else acabou := verdadeiro;
18            end
19       t_r := tupla para qual pr aponta;
20       while (pr ≠ nulo and t_r[AtribJunção] < t_s[AtribJunção]) do
21            begin
22                 configure pr para apontar para a próxima tupla de r;
23                 t_r := tupla para qual pr aponta;
24            end
25       while (pr ≠ nulo and t_r[AtribJunção] = t_s[AtribJunção]) do
26            begin
27                 for each t_s in S_s do
28                      begin
29                           adicione t_s ⋈ t_r ao resultado;
30                      end
31                 configure pr para apontar para a próxima tupla de r;
32                 t_r := tupla para qual pr aponta;
33            end
34   end
```

Figura 6: Junção *merge*.

3.2 ESTRATÉGIAS PARA EXECUÇÃO DO ALGORITMO DE JUNÇÃO MERGE ENTRE TABELAS WEB HETEROGÊNEAS

Através de uma amostra com 1.000 tabelas extraídas da Web no domínio de filmes, foi feita uma análise estrutural e verificou-se que o caso heterogêneo mais frequente é de tabelas que apresentam as linhas transpostas, ou seja, os rótulos são dispostos na primeira coluna, e não na primeira linha. Assim, escolheu-se essa estrutura para a realização de experimentos preliminares com o objetivo de determinar a forma menos custosa, em termos computacionais, de lidar com tabelas Web heterogêneas.

O objetivo destes experimentos preliminares foi determinar a melhor

forma de executar o algoritmo de junção *merge* da Figura 6 entre tabelas transpostas. A extração e a forma como o algoritmo de junção é executado diferenciam três as estratégias definidas. Na primeira, a tabela foi extraída do formato HTML sem alterar sua estrutura, para posterior execução de um método que a transpõe. Assim, a tabela estará no formato tradicional e o algoritmo de junção *merge* pode ser aplicado sem alterações. A segunda abordagem trabalha com a tabela em seu formato original. Para isso, o algoritmo de junção *merge* foi adaptado para interpretar colunas como linhas. Já a terceira, realiza a transposição da tabela durante sua extração da Web. As subseções a seguir detalham cada uma das estratégias, bem como os algoritmos utilizados.

3.2.1 Abordagem 1: Transposição de tabela após extração

Na primeira estratégia implementada, as tabelas tradicionais e transpostas são extraídas da mesma forma, conforme o algoritmo presente na Figura 7. O arquivo com a tabela HTML é percorrido linha a linha e a cada *tag* de linha encontrada (<*tr*>), uma nova *LinkedList*<*String*> é adicionada à estrutura principal. Na sequência, a cada *tag* de coluna encontrada (<*td*>), uma nova *String* é adicionada à estrutura referente à linha em questão. Dessa forma, não se altera a estrutura das tabelas originais. Para realizar a junção *merge*, primeiramente a tabela que não se encontra no formato tradicional é transposta conforme o algoritmo da Figura 8 de complexidade $O(numlinhas * numcolunas)$.

3.2.2 Abordagem 2: Transposição de tabela durante extração

Esta abordagem extrai as tabelas da mesma forma descrita anteriormente, porém não realiza a transposição. Para suportar tabelas transpostas, foi implementado um método que retorna determinada coluna como se fosse linha, conforme a Figura 9. Este método é executado durante o algoritmo de junção *merge* a cada vez que *pr* ou *ps*, dependendo de qual tabela está transposta, são configuradas para apontar para a próxima tupla (linhas 1, 2, 7, 15, 22 e 31 da Figura 6). O método possui complexidade $O(n)$ e é executado uma vez para cada coluna da tabela, ou seja, n vezes. Dessa forma, o aumento de complexidade total em relação à junção de tabelas tradicionais é $O(numlinhas * numcolunas)$.

```
entrada := primeira linha do arquivo HTML;
i := 0;
while (entrada ≠ nulo) do
    begin
        if (entrada ⊃ "<tr>")
            then begin
                linhaTmp := nulo;
                fimDeLinha := falso;
                j := 0;
                while (not fimDeLinha) do
                    begin
                        if (entrada ⊃ "<td>")
                            then begin
                                linhaTmp[j] := valor em entrada
entre "<td>" e "</td>";
                                j := j + 1;
                            end
                        entrada := próxima linha do arquivo HTML;
                        if (entrada ⊃ "</tr>")
                            then begin
                                fimDeLinha := verdadeiro;
                                tabela[i] := linhaTmp;
                            end
                    end
                i := i + 1;
            end
        entrada := próxima linha do arquivo HTML;
    end
```

Figura 7: Algoritmo para extração da tabela HTML.

```
numColunas := número de colunas da tabela invertida;
numLinhas := número de linhas da tabela invertida;
for i := 0 until numColunas do
    begin
        for j := 0 until numLinhas do
            begin
                linhaTmp[j] := tabelaInvertida[j][i];
                j := j + 1;
            end
        tabelaTradicional[i] := linhaTmp;
        i := i + 1;
    end
```

Figura 8: Algoritmo de transposição de tabela.

```
numLinhas := número total de linhas da tabela;
j := índice da coluna que se deseja transpor;
for i := 0 until numLinhas do
    begin
        colunaTransposta[i] := tabelaInvertida[i][j];
        i := i + 1;
    end
```

Figura 9: Algoritmo que obtém determinada coluna como linha.

3.2.3 Abordagem 3: Adaptação do algoritmo de junção *merge*

A diferença desta estratégia está na extração das tabelas transpostas. O algoritmo está exposto na Figura 10. Antes de alocar a estrutura para arma-

zenamento do conteúdo da tabela, conta-se quantas *tags* de linha e de coluna o arquivo possui. Este trecho difere a abordagem das anteriores e possui complexidade $O(n)$. Após, percorre-se o arquivo para extrair seu conteúdo, com uma iteração a cada linha. O primeiro valor é salvo na primeira posição da primeira *LinkedList*<*String*>; o segundo, na primeira posição da segunda *LinkedList*<*String*> e assim por diante, até percorrer toda a tabela do arquivo. Dessa forma, as tabelas transpostas são salvas com a mesma estrutura de uma tabela tradicional e o algoritmo de junção *merge* pode ser executado sem alterações.

3.3 EXPERIMENTOS

Foram realizados experimentos para avaliação de desempenho com as três abordagens apresentadas na Seção 3.2. O conjunto de dados para os testes consistiu em 20 tabelas HTML sobre o domínio de filmes extraídas da Web. Para testar a aplicação do algoritmo no pior caso, onde toda tupla encontra correspondente para junção, as tabelas foram alteradas e tiveram seu número de linhas aumentado. Dessa forma, as abordagens foram testadas com tabelas que possuíam entre 1, 250, 500 ou 1.000 tuplas com valores iguais para o atributo utilizado na junção ("*Title*"). Foram testados casos onde nenhuma, uma e as duas tabelas encontravam-se transpostas, sempre com 50.000 repetições para observar diferenças significativas no tempo de execução.

A Figura 11 apresenta os gráficos com o tempo de execução, medido em segundos, em função do número de tuplas com valor equivalente no atributo considerado para a junção. Percebe-se que a abordagem com maior custo computacional é a segunda, onde o algoritmo de junção *merge* foi modificado. Com isso, decidiu-se realizar o pré-processamento de tabelas Web heterogêneas antes de usá-las como base para consultas, em vez de adaptar os operadores relacionais a elas.

```
numLinhas := 0;
numColunas := 0;
entrada := primeira linha do arquivo HTML;
while (entrada ≠ nulo) do
    begin
        if (entrada ⊃ “<tr>”)
            then begin
                fimDeLinha := falso;
                while (numLinhas = 0 and not fimDeLinha) do
                    begin
                        if (entrada ⊃ “<td>”)
                            then begin
                                numColunas := numColunas + 1;
                            end
                        entrada := próxima linha do arquivo HTML;
                        if (entrada ⊃ “</tr>”)
                            then begin
                                fimDeLinha := verdadeiro;
                                numLinhas := numLinhas + 1;
                            end
                    end
            end
        entrada := próxima linha do arquivo HTML;
    end
entrada := primeira linha do arquivo HTML;
i := 0;
while (entrada ≠ nulo) do
    begin
        if (entrada ⊃ “<tr>”)
            then begin
                fimDeLinha := falso;
                j := 0;
                while (not fimDeLinha) do
                    begin
                        if (entrada ⊃ “<td>”)
                            then begin
                                linha := nulo;
                                for k := 0 until numLinhas do
                                    begin
                                        linha[k] := nulo;
                                    end
                                tabela[j] := linha;
                                coluna := valor entre “<td>” e “</td>”
em entrada;
                                tabela[j][i] := coluna;
                                j := j + 1;
                            end
                        entrada := próxima linha do arquivo HTML;
                        if (entrada ⊃ “</tr>”)
                            then begin
                                fimDeLinha := verdadeiro;
                            end
                    end
                    i := i + 1;
            end
    end
```

Figura 10: Algoritmo que transpõe a tabela durante sua extração.

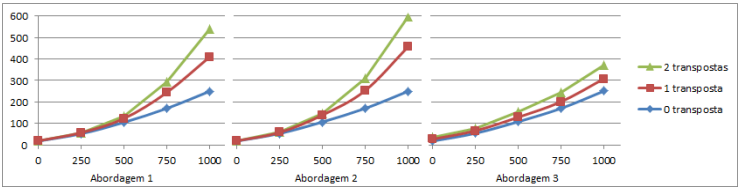


Figura 11: Gráficos com o tempo de execução em função do número de linhas das tabelas utilizadas para as três abordagens.

4 WT2RT: TAXONOMIA, FORMALIZAÇÃO E ALGORITMOS PARA UNIFICAÇÃO DE WEBTABLES

Dada a grande variedade estrutural de tabelas Web observada na análise da amostra inicial, percebeu-se a necessidade de entender melhor cada um dos tipos, principalmente os que possuem dados relacionais de qualidade. Com isso, modificou-se o foco desta dissertação de modo a catalogar os casos mais representativos de tabelas Web e então trazê-los para a estrutura conhecida nos bancos de dados relacionais.

Este capítulo apresenta a abordagem proposta nesse trabalho, intitulada WT2RT (*Web Table to Relational Table*). A Seção 4.1 traz uma taxonomia para tabelas Web heterogêneas. Através das categorias definidas, a Seção 4.2 descreve um classificador baseado em Redes Neurais Artificiais e a Seção 4.3 propõe algoritmos para unificação estrutural das tabelas Web. Por fim, é feita uma comparação com trabalhos relacionados na Seção 4.4.

4.1 CLASSIFICAÇÃO DE TABELAS WEB

A primeira parte desta seção traz conceitos gerais, na Subseção 4.1.1, importantes para a definição apropriada de cada categoria, na Subseção 4.1.2. Já as tabelas Web usadas apenas para leiaute serão descritas informalmente na Subseção 4.1.3.

O Modelo Relacional introduzido por Codd (CODD, 1970) considera *relações* como estruturas de dados. Dados n conjuntos de domínios $S_1, \cdots, S_n$, R é uma relação nestes n conjuntos se é um conjunto de n-tuplas, sendo que o primeiro elemento de cada tupla vem de S_1, o segundo de S_2, e assim por diante (CODD, 1970). Tendo em vista essa definição, são propostas as definições subsequentes para formalizar tabelas Web através de uma representação de dados relacionais. Como elas tem propósito relacional, devem ser tratadas como relações.

4.1.1 Conceitos Gerais

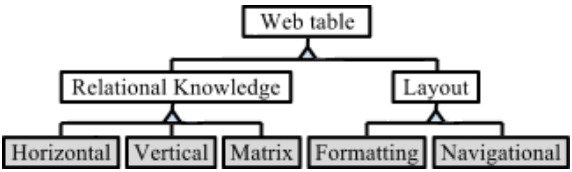
A estrutura apresentada na Tabela 1, com x linhas e y colunas, é usada para ilustrar alguns conceitos.

Definição 4.1.1 (Web Table)**.** *Denominam-se* WEB TABLES WT *estruturas encontradas em páginas Web delimitadas por tags <table>, em HTML, e compostas por um conjunto ordenado de x linhas e y colunas.*

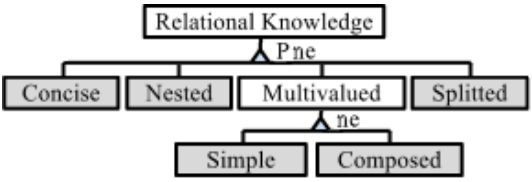
Tabela 1: Estrutura de uma Web table.

v_{11}	v_{12}	$\cdots$	v_{1y}
v_{21}	v_{22}	$\cdots$	v_{2y}
$\vdots$	$\vdots$		$\vdots$
v_{x1}	v_{x2}	$\cdots$	v_{xy}

Definição 4.1.2 (Cell, Label, Data *e* Multivalued Data)**.** *Seja* WT *uma* WEB TABLE *com x linhas e y colunas, representadas na Tabela 1. Cada intersecção entre uma linha e uma coluna determina uma* CELL c_{ij}*, a qual possui valor* v_{ij}*, para* $\forall\ i$ *e* $\forall\ j$*. Valores* v_{ij} *vêm do conjunto* $L = \{l_1, ..., l_y\}$*, composto por* LABELS*, ou do conjunto* $D = \{d_1, ..., d_{(x-1)\cdot y}\}$*, composto por* DATA*. O domínio dos elementos em L é string, enquanto o domínio dos elementos em D pode ser string,* WEB TABLES*, valores nulos ou um conjunto de valores atômicos. Um valor* v_{ij} *é dito* MULTIVALUED DATA *sse for composto por mais de um valor* DATA*.*



(a) Classificação Principal.



(b) Classificação Secundária.

Figura 12: Classificação hierárquica dos tipos de formatação de Web tables.

Comparando com o Modelo Relacional (CODD, 1970), LABELS equivalem aos nomes de domínios, enquanto DATA correspondem a elemen-

tos destes domínios. De acordo com a notação apresentada em (ABITEBOUL; HULL; VIANU, 1995), LABELS correspondem a *atributos*. A maior diferença entre WEB TABLES e Tabelas Relacionais é que a segunda possui uma estrutura única, na qual a primeira linha é formada pelos nomes dos atributos; e as outras, pelas tuplas com valores ordenados para os atributos acima. Pode-se ver um exemplo de WEB TABLE na Figura 13(a), onde os LABELS (*Title*, *Year* e *Genre*), localizados na primeira linha, correspondem aos nomes dos atributos para os valores DATA das linhas abaixo.

A fim de entender melhor algumas características das WEB TABLES, é importante introduzir o conceito de SEQUENTIAL CELLS. Ele será usado na definição de MERGED CELLS, presentes em algumas categorias apresentadas depois.

Definição 4.1.3 (Sequential Cells)**.** *Sejam c_{ab} e c_{de} duas* CELLS *de uma* WEB TABLE WT*, onde a e d são índices para as linhas, e b e e, para as colunas. As* CELLS *c_{ab} e c_{de} são ditas* SEQUENTIAL CELLS *sse apenas uma das seguintes condições for verdadeira: $d = a + 1$ ou $e = b + 1$.*

Sejam m, n e p três CELLS arbitrárias. Se m for sequencial a n e n for sequencial a p, então m, n e p compõem um conjunto de SEQUENTIAL CELLS. Por exemplo, na Figura 13(a), observe CELLS na primeira ROW. A que contém o valor *Title* é sequencial a CELL com valor *Year*, e esta segunda é sequencial a CELL com valor *Genre*. Juntas, as três compõem um conjunto de SEQUENTIAL CELLS.

Definição 4.1.4 (Merged Cell)**.** *Seja* MC *um conjunto de* SEQUENTIAL CELLS *de uma* WEB TABLE WT. MC *é dito* MERGED CELL *sse todos os seus elementos estão associados ao mesmo valor v, onde $v \in L$ ou $v \in D$.*

Um exemplo de MERGED CELL pode ser visto na Figura 15, onde os valores v_{21}, v_{22} e v_{23} estão todos associados a *SHRUBS*. Como eles têm o mesmo valor, são apresentadas em uma única CELL. É importante observar que apenas SEQUENTIAL CELLS podem formar uma MERGED CELL.

4.1.2 Relational Knowledge Web Tables

Considerando todas as definições apresentadas, agora são introduzidas e categorizadas as estruturas de WEB TABLE. A Figura 12 mostra uma classificação hierárquica delas[1]. As estruturas pertencentes a alguma categoria *Relational Knowledge* na Classificação Principal também passam pela Classificação Secundária. Note que, como as tabelas com dados relacionais

[1] Classificação inspirada em (CRESTAN; PANTEL, 2011)

são o foco deste trabalho, apenas estas categorias são formalizadas. As outras são informalmente descritas na Subseção 4.1.3.

Definição 4.1.5 (Horizontal Web Table)**.** *Uma* WEB TABLE WT *é dita* HORIZONTAL *sse os valores* v_{ij} *(*$b \leqslant i \leqslant x$*;* $j = 1$*) vêm do domínio* S_1*, os valores* v_{ij} *(*$b \leqslant i \leqslant x$*;* $j = 2$*), do domínio* S_2*, e assim por diante.* b *corresponde ao índice da primeira linha em que* $\exists\, v_{ij} \in D\ (\forall\, i; \forall\, j)$*. Se* $\exists\, v_{ij} \in L\ (i = 1; \forall\, j)$*,* b *assume valor* 2 *e* $v_{ij} \in D\ (b \leqslant i \leqslant x; \forall\, j)$*. Caso contrário,* $b = 1$ *e* $v_{ij} \in D\ (\forall\, i; \forall\, j)$*.*

Tendo em vista a definição de Codd (CODD, 1970) para o Modelo Relacional, os LABELS correspondem aos nomes dos domínios em que a WEB TABLE WT foi definida. Como exemplificado na Figura 13(a), o conjunto de LABELS localiza-se na primeira linha (*Title*, *Year* e *Genre*), e os valores das linhas restantes são DATA, como em uma tabela relacional.

Title	Year	Genre
The Freshman	1925	Comedy
Maker of Men	1931	Drama
Horse Feathers	1932	Comedy
College Coach	1933	Drama
Pigskin Parade	1936	Comedy

(a) Horizontal Web table.

Robert De Niro	
Born	August 17, 1943 New York, NY
Nationality	American
Occupation	Actor, director and producer
Years active	1959–present

(b) Vertical Web table.

Figura 13: Exemplos de Web tables

Definição 4.1.6 (Vertical Web Table)**.** *Uma* WEB TABLE WT *é dita* VERTICAL *sse os valores* v_{ij} *(*$i = 1$*;* $b \leqslant j \leqslant y$*) vêm do domínio* S_1*, os valores* v_{ij} *(*$i = 2$*;* $b \leqslant i \leqslant y$*), do domínio* S_2*, e assim por diante.* b *corresponde ao índice da primeira coluna em que* $\exists\, v_{ij} \in D\ (\forall\, i; \forall\, j)$*. Se* $\exists\, v_{ij} \in L\ (\forall\, i; \forall\, j)$*,* $b = 2$*,* $v_{ij} \in L\ (\forall\, i;\ j = 1)$ *e* $v_{ij} \in D\ (\forall\, i;\ b \leqslant j \leqslant y)$*. Caso contrário,* $b = 1$ *e* $v_{ij} \in D\ (\forall\, i; \forall\, j)$*.*

Em outras palavras, WT é dita VERTICAL se suas tuplas estão dispostas verticalmente. O conjunto de LABELS, quando presente, localiza-se na primeira coluna. Na estrutura representada na Tabela 1, o LABEL representado por v_{11} estaria associado ao DATA localizado na primeira linha, i.e., de v_{12} até v_{1n}. Esta estrutura foi informalmente definida como *Horizontal Listing* em (CRESTAN; PANTEL, 2011) e como *Vertical Table* em (VENETIS

et al., 2011). A Figura 13(b) mostra um exemplo dessa WEB TABLE, onde os LABELS (*Born*, *Residence*, *Nationality*, etc.) estão localizados na primeira coluna.

Definição 4.1.7 (Matrix Web Table)**.** *Sejam S_1, S_2 e S_3 três domínios diferentes. Uma* WEB TABLE WT *é dita* MATRIX *sse os valores $v_{ij} \in S_1$ ($i = 1$; $2 \leqslant j \leqslant y$), $v_{ij} \in S_2$ ($2 \leqslant i \leqslant x$; $j = 1$) e $v_{ij} \in S_3$ ($2 \leqslant i \leqslant x$; $2 \leqslant j \leqslant y$). Seja V um conjunto com todos os valores $v_{ij} \in$ WT. $v_{11} \in L$ e $(V - v_{11}) \in D$. Cada valor v_{ij} ($2 \leqslant i \leqslant x$; $2 \leqslant j \leqslant y$) pertence ao mesmo objeto que v_{mj} ($m = 1$) e v_{in} ($n = 1$).*

Em outras palavras, cada valor que não está na primeira linha/coluna é associado a um valor no cabeçalho da linha e a outro no cabeçalho da coluna. A Figura 14 mostra estatísticas para acidentes de carros, onde o domínio S_1 é *década*, S_2 é *causa* e S_3 é o *número de acidentes*. Uma pessoa facilmente percebe que o conteúdo da CELL c_{22}, o número 26, não é uma instância do cabeçalho da linha (*1980s*) nem da coluna (*Pilot error*), já que eles não são LABELS. Este valor corresponde ao *número de acidentes* que aconteceu na década de *1980s* por *Pilot error*. O valor na CELL c_{11} é o único LABEL nessa WEB TABLE, e nesse caso, está associado aos valores da primeira coluna. Não estão presentes, na tabela, LABELS para os valores de *década* nem para os *números de acidentes*.

Cause	1980s	1990s	2000s
Pilot Error	26	27	30
Weather	14	10	8
Mechanical Failure	20	18	24

Figura 14: Exemplo de Matrix Web table.

Definição 4.1.8 (Concise Web Table)**.** *Seja* MC *uma* MERGED CELL *qualquer. Uma* WEB TABLE WT *é dita* CONCISE *sse* WT $\supset$ MC.

Em uma CONCISE WEB TABLE, há ocorrência de MERGED CELLS para evitar repetição de valores, de forma que a WEB TABLE se torne mais compacta, i.e., concisa. Desafios na interpretação desta estrutura já foram mencionados em um trabalho recente (VENETIS et al., 2011), onde a MERGED CELL é considerada um *sub-header* para as linhas abaixo dela. Neste trabalho, este caso foi abordado de forma mais geral. No caso ilustrado na Figura 15, os valores da MERGED CELL representam DATA em comum para as linhas abaixo e podem ser interpretados como *sub-headers*. Também podem ser representados como um novo atributo, em uma nova coluna, com o valor

shrubs para as plantas *azalea* e *buddleia*; e *cultivated annuals* para as plantas *alyssum* e *candytuft*.

PLANT	COLOR	HEIGHT
SHRUBS		
Azalea	variable	shrub
Buddleia	blue, pink, white	shrub
CULTIVATED ANNUALS		
Alyssum	violet, white	4 inches
Candytuft	white, pink	8-10 inches

Figura 15: Exemplo de Concise Web table onde as *merged cells* atuam como *sub-headers*.

Entretanto, na situação da Figura 16, as CELLS dispostas verticalmente foram mescladas e não atuam como *sub-headers*. De modo a não repetir valores de para *Year*, o qual seria o mesmo para os filmes *Death at a Funeral*, *I Love You Too* e *Pete Smalls is Dead* (*2010*), as CELLS originais foram mescladas em uma só.

Year	Title	Role
2010	*Death at a Funeral*	Frank
	I Love You Too	Charlie
	Pete Smalls Is Dead	KC
2011	*A Little Bit of Heaven*	Vinnie

Figura 16: Exemplo de Concise Web table onde as *merged cells* evitam repetição de valores.

Definição 4.1.9 (Nested Web Table)**.** *Seja* $M = \{T_1, ..., T_n\}$ *um conjunto de* WEB TABLES. *Uma* WEB TABLE WT *é dita* NESTED *sse for composta por dois ou mais elementos de M.*

A WEB TABLE apresentada na Figura 17 é classificada como NESTED, além de ser um exemplo de VERTICAL. Pode ser observado que há duas WEB TABLES (*General information* and *Broadcast*) aninhadas.

Definição 4.1.10 (Splitted Web Table)**.** *Uma* WEB TABLE WT *é dita* SPLITTED *sse seus* LABELS *apresentam repetições ordenadas sequenciais no cabeçalho. Seja s o número dessas repetições. Consequentemente, cada* LABEL *é repetido a cada* $\frac{z}{s+1}$ CELL(S)*, onde* $z = x$ *se a* SPLITTED WEB TABLE *for* HORIZONTAL*;* $e\ z = y$ *se for* VERTICAL.

Comparando com o modelo Relacional, pode-se dizer que cada linha de DATA de uma SPLITTED WEB TABLE, é composta por $s+1$ tuplas. Este fato pode ser observado na Figura 18, onde a SPLITTED WEB TABLE tem $s = 1$, i.e., foi partida horizontalmente uma vez. Dessa forma, o conjunto de LABELS (*Rank*, *City name* e *Pop.*) aparece repetido uma vez. Na analogia com o Modelo Relacional, pode-se dizer que as linhas de 2 a 6 são compostas por duas tuplas.

Definição 4.1.11 (Multivalued Web Table)**.** *Seja v_{ij} um valor qualquer de uma* WEB TABLE WT. WT *é dita* MULTIVALUED *sse* $\exists\, v_{ij} \in$ WT *que é* MULTIVALUED DATA*, composto por um conjunto de k valores* DATA $\{m_1, ..., m_k\}$*, os quais vêm dos domínios* $\{S_1, ..., S_k\}$*, respectivamente.*

Neste caso, a WEB TABLE WT possui valores DATA que são conjuntos de outros valores DATA. Por exemplo, para representar atores de um filme, seus nomes compõem um conjunto de valores, que será o valor para a CELL relativa ao elenco.

Definição 4.1.12 (Simple Multivalued Web Table)**.** *Seja v_{ij} um* MULTIVALUED DATA*, de uma* WEB TABLE WT*, com k elementos.* WT *é dita* SIMPLE MULTIVALUED *sse* $S_1 = ... = S_k$*, i.e., todos os k elementos de v_{ij} vêm do mesmo domínio.*

A WEB TABLE da Figura 17 possui MULTIVALUED DATA para o elenco, em *Starring*. O valor DATA correspondente é formado por quatro nomes pertencentes ao mesmo domínio: ator.

General information	
Title	The IT Crowd
Created by	Graham Linehan
Starring	Chris O'Dowd Richard Ayoade Katherine Parkinson Matt Berry
Broadcast	
Picture format	576i (16:9 SDTV)
Audio format	Stereo
Original run	3 February 2006 – 27 September 2013

Figura 17: Exemplo de Nested Web table.

Rank	City name	Pop.	Rank	City name	Pop.
1	São Paulo	11,316,149	6	Belo Horizonte	2,385,639
2	Rio de Janeiro	6,355,949	7	Manaus	1,832,423
3	Salvador	3,093,605	8	Curitiba	1,764,540
4	Brasília	2,609,997	9	Recife	1,536,934
5	Fortaleza	2,476,589	10	Porto Alegre	1,413,094

Figura 18: Exemplo de Splitted Web table.

Definição 4.1.13 (Composed Multivalued Web Table)**.** *Seja* v_{ij} *um* MULTIVALUED DATA *de uma* WEB TABLE WT. WT *é dita* COMPOSED MULTIVALUED *sse* $S_1 \neq ... \neq S_k$*, i.e., todos os k elementos de* v_{ij} *vêm de domínios diferentes.*

A WEB TABLE da Figura 13(b) tem COMPOSED MULTIVALUED DATA no valor de *Born*, o qual consiste em informações sobre a data e a localização do nascimento de Robert De Niro.

4.1.3 Layout Web Tables

A maioria das tabelas HTML encontradas na Web são usadas para leiaute. De acordo com (CRESTAN; PANTEL, 2011), estas são divididas em FORMATTING e NAVIGATIONAL, exemplificadas na Figura 19. A primeira é usada para organizar os elementos (texto, imagens, vídeos, tabelas, etc.) na página. Por outro lado, a segunda categoria dispõe os itens de menus, com *links*, para facilitar a navegação.

4.2 WTCLASSIFIER, UM CLASSIFICADOR BASEADO EM REDES NEURAIS ARTIFICIAIS

Diante da grande quantidade de dados presente na Web, é imprescindível automatizar o processo de classificação. Em (CRESTAN; PANTEL, 2011), foi usado um classificador baseado em Árvore de Decisão, que analisa características de leiaute e conteúdo para aprender cada categoria de WEB TABLE. Neste trabalho, foi desenvolvido o WTCLASSIFIER com o propósito de categorizar tabelas Web dentro da taxonomia apresentada na Seção 4.1. Para o processo de aprendizagem, foi usada uma Rede Neural Artificial pela sua resiliência, fundamental a conjuntos de dados tão variados como os extraídos da Web (Seção 2.1). O funcionamento do classificador proposto é apresen-

Figura 19: Exemplo de Navigational e Formatting Web table.

tado na Subseção 4.2.1, enquanto a implementação é descrita na Subseção 4.2.2.

4.2.1 Funcionamento básico do WTClassifier

O funcionamento do classificador proposto consiste em analisar características de tabelas Web cujas categorias originais são conhecidas para aprender a partir delas. Os processos de busca e extração de tabelas da Web estão fora do escopo desta dissertação. As seções a seguir detalham seu funcionamento.

Como forma de aprendizagem, foi utilizada uma Rede Neural Artificial do tipo *Multilayer Perceptron*, e como dados de entrada, as vinte características listadas na Subseção 2.3.2 e cinco adicionais:

1. Posição de tabelas internas, quando presentes: se alguma célula possui, em seu interior, uma *tag* HTML *<table>*, essa característica é representada pelos números de linha e coluna dessa CELL.
2. Fração de células contendo listas ordenadas: porcentagem de células contendo *tag* HTML *<ol>*.
3. Fração de células contendo listas não-ordenadas: porcentagem de células contendo *tag* HTML *<ul>*.
4. Fração de células contendo vírgulas: porcentagem de células contendo pelo menos um caractere de vírgula.
5. Fração de células contendo parênteses: porcentagem de células con-

tendo pelo menos um par de parênteses.

A primeira característica auxilia na identificação de FORMATTING WEB TABLE, enquanto as outras quatro geralmente caracterizam a presença de MULTIVALUED DATA. No total, são vinte e cinco dados de entrada. As saídas consistem em números reais entre 0 e 1, indicando a probabilidade da tabela em questão pertencer a cada categoria.

Após coletadas as informações de entrada, estas precisam ser normalizadas para evitar que a função de ativação dos neurônios seja saturada. Esse processo é feito utilizando a equação

$$valor = \frac{valor - valorMin}{valorMax} \tag{4.1}$$

onde *valorMin* representa o menor valor de cada característica, e *valorMax*, o maior. Dessa forma, para cada característica do conjunto de treino, o maior valor foi setado em 1, e todos os outros foram relativizados proporcionalmente.

Para o processo de aprendizagem supervisionada, foi utilizado um conjunto com 4.000 tabelas Web manualmente rotulado de acordo com as categorias apresentadas na Seção 4.1. O WTCLASSIFIER aprende, então, analisando os padrões de cada categoria, representado na lista de características citada anteriormente, similar à abordagem usada em (CRESTAN; PANTEL, 2011).

4.2.2 Implementação

Foram feitas duas redes neurais utilizando o *framework Neuroph*[2], sendo uma para cada Classificação (Principal e Secundária). Em ambas, o algoritmo de aprendizagem que se mostrou mais eficiente foi o de *Resilient Backpropagation*.

Para a Classificação Principal, as vinte e cinco características listadas anteriormente (Subseções 2.3.2 e 4.2.1) representaram os dados de entrada. Como saída, cinco neurônios indicando as categorias: HORIZONTAL, VERTICAL, MATRIX, FORMATTING e NAVIGATIONAL. Essas classes são mutuamente exclusivas, i.e., uma tabela não pode estar em mais de uma ao mesmo tempo. A camada escondida continha trinta neurônios.

Já na Classificação Secundária, além das vinte e cinco características extraídas das tabelas, foi utilizada a categoria obtida na Classificação Principal como dado de entrada, representada através de três neurônios: um para

[2]Framework para criação de Redes Neurais http://neuroph.sourceforge.net

HORIZONTAL, um para VERTICAL e um para MATRIX, pois apenas essas, também chamadas de Relational Knowledge Web Tables, passaram para esse segundo classificador. Sua saída consistiu em cinco neurônios, representando suas cinco categorias secundárias não mutuamente exclusivas, i.e., uma tabela pode ser classificada em mais de uma: CONCISE, NESTED, SPLITTED, SIMPLE MULTIVALUED e COMPOSED MULTIVALUED. A camada escondida continha trinta e cinco neurônios.

Para prever o erro das redes foi realizado o processo de *stratified tenfold cross validation*. Os dados são divididos randomicamente em 10 partes nas quais cada categoria é representada em aproximadamente a mesma proporção que se apresenta no conjunto completo. Cada parte é deixada de lado uma vez, enquanto a aprendizagem é feita com as outras nove partes. Após, o erro é medido no grupo que ficou de fora. Esse processo de aprendizagem é executado com cada um dos conjuntos de treino diferentes, totalizando dez vezes. Finalmente, é calculada a média entre as dez estimativas de erro para estimar da taxa de erro real. Estudos em diferentes conjuntos de dados, com diferentes técnicas de aprendizagem, mostraram que 10 é o número ideal de partes para conseguir a melhor estimativa de erro (WITTEN; FRANK, 2005).

4.3 ALGORITMOS PARA UNIFICAÇÃO ESTRUTURAL

De modo a facilitar a interpretação das WEB TABLES detalhadas anteriormente, elas devem ser apresentadas em uma estrutura homogênea. Apesar de trabalhos recentes (CRESTAN; PANTEL, 2011) mencionarem desafios de extrair conhecimento de algumas categorias de WEB TABLES, eles não descreveram nenhuma regra para o processo de extração.

Através de uma análise das estruturas coletadas por um *Web crawler*, foram definidos algoritmos que se aplicam à maioria dos casos encontrados. Eles descrevem como trazer tabelas heterogêneas para o formato do Modelo Relacional e não foram implementados nesta dissertação. As seções seguintes apresentam algoritmos para cada categoria da taxonomia formalmente definida na Subseção 4.1.2, exceto a HORIZONTAL WEB TABLE, que já se encontra no formato relacional.

4.3.1 Vertical Web Table

Quando uma tabela encontra-se vertical, deve ser transposta de modo a dispor suas tuplas horizontalmente. O processo de transposição consiste em

converter colunas em linhas, de acordo com o Algoritmo 1. Em cada iteração (linha 4), as linhas da tabela original são percorridas e têm seus valores copiados para uma estrutura temporária (linha 6), que irá compor a tupla. Após (linha 8), a tupla devidamente transposta é copiada para a tabela relacional destino.

Terminado esse processo, a WEB TABLE da Figura 13(b) torna-se a representada na Figura 20. Há, ainda, três casos de dados multivalorados nesta WEB TABLE (valores DATA correspondentes aos LABELS *Born* e *Occupation*). Seus tratamentos são descritos nas Subseções 4.3.6 e 4.3.7.

Algoritmo 1 Algoritmo para converter uma Vertical Web table Vertical em Tabela Relacional.

```
1:  procedure VERTICALLISTING(verticalTable)
2:      numRows ← number of verticalTable rows
3:      numColumns ← number of verticalTable columns
4:      for i ← 1 until numColumns do
5:          for j ← 1 until numRows do
6:              line[j] ← verticalTable[j][i]
7:          end for
8:          relationalTable[i] ← line
9:      end for
10:     return relationalTable
11: end procedure
```

Born1	Born2	Born3	Born4	Born5	Nationality	Occupation	Years active1	Years active2
August	17	1943	New York	NY	American	Actor	1959	present
August	17	1943	New York	NY	American	director	1959	present
August	17	1943	New York	NY	American	producer	1959	present

Figura 20: Vertical Web table representada como Tabela Relacional.

4.3.2 Matrix Web Table

Na estrutura matricial, todos os valores que não se encontram na primeira linha/coluna devem ser dispostos em uma única coluna, já que pertencem ao mesmo domínio. Dessa forma, haverá repetição dos valores originalmente presentes nos cabeçalhos lateral e superior.

O Algoritmo 2 descreve os passos para transformar uma MATRIX WEB TABLE em Tabela Relacional. A CELL c_{11} (linha 2) é extraída como LA-

BEL dos demais valores da primeira coluna (linha 11). Conforme a Definição 4.1.7, valores na primeira linha são do mesmo domínio. Portanto, eles são transpostos e formam a segunda coluna da nova tabela (linha 12). A terceira é uma combinação dos valores das CELLS c_{ij} ($2 \leqslant i \leqslant x$; $2 \leqslant j \leqslant y$) (linha 13). Esse processo aplicado à tabela da Figura 14 resulta na tabela da Figura 21.

Algoritmo 2 Algoritmo para converter uma Matrix Web table em Tabela Relacional.

```
 1: procedure MATRIX(matrixTable)
 2:     relationalTable[1][1] ← matrixTable[1][1]
 3:     relationalTable[1][2] ← "attribute1"
 4:     relationalTable[1][3] ← "attribute2"
 5:     rowCounter ← 0
 6:     numRows ← number of matrixTable rows
 7:     numColumns ← number of matrixTable columns
 8:     for i ← 2 until numRows do
 9:         row ← matrixTable[i]
10:         for j ← 2 until numColumns do
11:             relationalTable[rowCounter][1] ← matrixTable[i][1]
12:             relationalTable[rowCounter][2] ← matrixTable[1][j]
13:             relationalTable[rowCounter][3] ← matrixTable[i][j]
14:             rowCounter++
15:         end for
16:     end for
17:     return relationalTable
18: end procedure
```

4.3.3 Concise Web Table

Tabelas concisas possuem três diferentes casos: (i) células horizontalmente mescladas ocupando toda a linha, (ii) células horizontalmente mescladas que não ocupam toda a linha e (iii) células mescladas horizontalmente. O Algoritmo 3 trata estes três casos.

Na primeira situação, a célula mesclada é extraída como uma nova coluna (linha 26) cujo valor é o mesmo para as linhas abaixo na tabela original. Após a execução deste Algoritmo, a tabela na Figura 15 se torna a representada na Figura 22.

Para o segundo e o terceiro casos, o valor na célula mesclada é copiado

Algoritmo 3 Algoritmo para converter uma Concise Web table em Tabela Relacional.

```
1: procedure CONCISE(conciseTable)
2:     numRows ← number of conciseTable rows
3:     numColumns ← number of conciseTable columns
4:     numFullColspan ← 0
5:     numRowspan ← inicializes all numColumns positions with 0
6:     for i ← 1 until numRows do
7:         numColspan ← 0
8:         isFullColspan ← whether all cells in the current row are merged
9:         if isFullColspan then
10:            lastFullColspanValue ← conciseTable[i][1]
11:            numFullColspan++
12:        else
13:            for j ← 1 until numColumns do
14:                isCellHorizontallyMerged ← whether cell
   conciseTable[i][j] is merged with conciseTable[i][j−1]
15:                isCellVerticallyMerged ← whether cell
   conciseTable[i][j] is merged with conciseTable[i−1][j]
16:                if isCellHorizontallyMerged then
17:                    numColspan++
18:                end if
19:                if isCellVerticallyMerged then
20:                    numRowspan[j]++
21:                end if
22:                rowCorrec ← i − numFullColspan − numRowspan[j]
23:                colCorrec ← j − numColspan
24:                relationalTable[i − numFullColspan][j] =
   conciseTable[rowCorrec][colCorrec]
25:                if isFullColspan > 0 then
26:                    relationalTable[i−numFullColspan][numColumns+
   1] = lastFullColspanValue
27:                end if
28:            end for
29:        end if
30:    end for
31:    return relationalTable
32: end procedure
```

Cause	attribute_1	attribute_2
Pilot Error	1980s	26
Pilot Error	1990s	27
Pilot Error	2000s	30
Weather	1980s	14
Weather	1990s	10
Weather	2000s	8
Mechanical Failure	1980s	20
Mechanical Failure	1990s	18
Mechanical Failure	2000s	24

Figura 21: Matrix Web table representada como Tabela Relacional.

PLANT	COLOR	HEIGHT	attribute_1
Azalea	variable	shrub	SHRUBS
Buddleia	blue, pink, white	shrub	SHRUBS
Alyssum	violet, white	4 inches	CULTIVATED ANNUALS
Candytuft	white, pink	8-10 inches	CULTIVATED ANNUALS

Figura 22: Concise Web table representada como Tabela Relacional.

para todas as células que a compõe (linha 24). A Figura 16 apresentam os valores (*2010*) repetidos para *Year*, como pode-se ver na Figura 23.

Year	Title	Role
2010	Death at a Funeral	Frank
2010	I Love You Too	Charlie
2010	Pete Smalls Is Dead	KC
2011	A Little Bit of Heaven	Vinnie

Figura 23: Concise Web table representada como Tabela Relacional.

4.3.4 Nested Web Table

WEB TABLES que se encontram aninhadas devem ser salvas separadamente. Por exemplo, as duas tabelas da Figura 17 são mostradas separadamente na Figura 24, após passarem pelo processo descrito no Algoritmo 4. Nesse caso, a tabela também possui MULTIVALUED DATA, cujo tratamento é descrito posteriormente, na Subseção 4.3.6.

Algoritmo 4 Pseudo código para converter uma Nested Web table em um conjunto de Tabelas Relacionais.

```
1:  procedure NESTED(nestedTable)
2:      numRows ← number of nestedTable rows
3:      numColumns ← number of nestedTable columns
4:      for i ← 1 until numRows do
5:          isNewTable ← whether the row i is the start of a new table
6:          if isNewTable & tableTmp ≠ null then
7:              add tableTmp to relationalTables
8:              tableTmp ← null
9:          end if
10:         add nestedTable[i] to tableTmp
11:     end for
12:     add tableTmp to relationalTables
13:     return relationalTables
14: end procedure
```

Title	Created by	Starring
The IT Crowd	Graham Linehan	Chris O'Dowd Richard Ayoade Katherine Parkinson Matt Berry

Picture format	Audio format	Original run
576i (16:9 SDTV)	Stereo	3 February 2006 – 27 September 2013

Figura 24: Nested Web table representada como duas Tabelas Relacionais.

4.3.5 Splitted Web Table

Em WEB TABLES que sofreram uma ou mais partições, primeiro é necessário determinar quantas vezes isso ocorreu e manter esse valor em s (Algoritmo 5, linha 4). O processo é feito contando quantas vezes o conjunto de LABELS é repetido no cabeçalho. Por exemplo, na Figura 18, o conjunto de LABELS é repetido uma vez, o que significa que a WEB TABLE foi partida uma vez. Então, percorre-se cada linha e cada $numColumns/s$ CELLS (linha 9), adiciona-se a tupla à Tabela Relacional (linha 10). Há uma exceção para a primeira linha, onde copia-se apenas os primeiros $numColumns/s$ valores, referentes aos LABELS (linha 12). Para a situação mencionada anteriormente, a Tabela Relacional equivalente é mostrada na Figure 25.

Algoritmo 5 Pseudo código para converter uma Splitted Web table em Tabela Relacional.

```
1:  procedure SPLITTED(splittedTable)
2:      numRows ← number of splittedTable rows
3:      numColumns ← number of splittedTable columns
4:      s ← number of times that the splittedTable is splitted
5:      for i ← 1 until numRows do
6:          countColumns ← 1
7:          for j ← 1 until numColumns do
8:              tuple[countColumns] ← splittedTable[i][j]
9:              if countColumns = numColumns/s then
10:                 add tuple to relationalTable
11:                 countColumns ← 1
12:                 if i = 1 then
13:                     break
14:                 end if
15:             else
16:                 countColumns++
17:             end if
18:         end for
19:     end for
20:     return relationalTable
21: end procedure
```

Rank	City name	Pop.
1	São Paulo	11,316,149
2	Rio de Janeiro	6,355,949
3	Salvador	3,093,605
4	Brasília	2,609,997
5	Fortaleza	2,476,589
6	Belo Horizonte	2,385,639
7	Manaus	1,832,423
8	Curitiba	1,764,540
9	Recife	1,536,934
10	Porto Alegre	1,413,094

Figura 25: Splitted Web table representada como Tabela Relacional.

4.3.6 Simple Multivalued Web Table

Nos casos de SIMPLE MULTIVALUED WEB TABLE, para cada linha contendo CELLS com MULTIVALUED DATA, precisa-se fazer o produto cartesiano entre todas as CELLS de sua linha (Algoritmo 6, linha 9). Dessa forma, são feitas todas as combinações entre os valores MULTIVALUED DATA e os outros DATA valores da mesma linha. Por exemplo, a WEB TABLE da Figura 13, com MULTIVALUED DATA para *Occupation*, tem esses valores separados em tuplas diferentes, como mostrado na Figura 20.

4.3.7 Composed Multivalued Web Table

Tendo em vista MULTIVALUED DATA compostos, seus elementos são separados em colunas diferentes (Algoritmo 7, linha 10). Um exemplo pode ser visto na Figura 20, onde *Born* é dividido em quatro colunas.

4.4 COMPARAÇÃO COM TRABALHOS RELACIONADOS A TABELAS WEB

A Tabela 2 lista quais trabalhos citaram quais categorias de tabelas Web. Nela, fica claro que as taxonomias ficaram mais abrangentes com o passar do tempo. Foi considerado que todos os que mencionaram HORIZONTAL e VERTICAL, também citaram *Attribute/Value* e *Enumeration*, já que são apenas casos específicos das primeiras. O mesmo vale para *Matrix* e *Calendar*.

Algoritmo 6 Pseudo código para converter uma Simple Multivalued Web table em Tabela Relacional.

```
1:  procedure SIMPLEMULTIVALUED(simpleMultivaluedTable)
2:      numRows ← number of simpleMultivaluedTable rows
3:      numColumns ← number of simpleMultivaluedTable columns
4:      for i ← 1 until numRows do
5:          splittedRow ← null
6:          for j ← 1 until numColumns do
7:              splittedRow[j] ← list of inclusive multivalued elements of
    simpleMultivaluedTable[i][j]
8:          end for
9:          cartesianProduct ← cartesian product among splittedRow ele-
    ments
10:         relationalTable ← cartesianProduct
11:     end for
12:     return relationalTable
13: end procedure
```

	Yoshida	Venetis	TabEx	WTClassifier
Horizontal	x	x	x	x
Vertical	x	x	x	x
Attribute/Value	x	x	x	x
Matrix			x	x
Calendar			x	x
Enumeration	x	x	x	x
Form			x	x
Formatting	x	x	x	x
Navigational			x	x
Concise		x		x
Nested				x
Simple Multiv.				x
Composed Multiv.				x
Splitted	x			x

Tabela 2: Comparação de categorias de tabelas mencionadas em trabalhos.

Algoritmo 7 Pseudo código para converter uma Composed Multivalued Web table em Tabela Relacional.

```
1: procedure COMPOSEDMULTIVALUED(composedMultivaluedTable)
2:     numRows ← number of composedMultivaluedTable rows
3:     numColumns ← number of composedMultivaluedTable columns
4:     columnShift ← 0
5:     isHeaderCorrected ← false
6:     for i ← 1 until numRows do
7:         for j ← 1 until numColumns do
8:             isComposedMultivaluedData ← whether composedMultivaluedTable[i][j] contains composed multivalued data
9:             if isComposedMultivaluedData then
10:                cellElements ← list of composed multivalued elements of composedMultivaluedTable[i][j]
11:                numMultivElements ← number of composed multivalued elements of composedMultivaluedTable[i][j]
12:                for k ← 1 until numMultivElements do
13:                    relationalTable[i][j + columnShift + k − 1] = cellElements[k]
14:                    if isHeaderCorrected = false then
15:                        relationalTable[1][j + columnShift + k − 1] ← concat k value to composedMultivaluedTable[1][j]
16:                    end if
17:                end for
18:                columnShift ← columnShift + numMultivElements − 1
19:            else
20:                relationalTable[i][j + columnShift] ← composedMultivaluedTable[i][j]
21:                if ¬isHeaderCorrected then
22:                    relationalTable[1][j + columnShift] ← composedMultivaluedTable[1][j]
23:                end if
24:            end if
25:        end for
26:        if i = 2 then
27:            isHeaderCorrected ← true
28:        end if
29:    end for
30:    return relationalTable
31: end procedure
```

	Yoshida	TabEx	WTClassifier
Propõe taxonomia	x	x	x
Número de categorias	9	10	10
Técnica de classificação	Max. Expectativa	Árvore de decisão	Rede neural
Formaliza categorias	não	não	sim
Nº de páginas visitadas	não citado	1.2B	175k
Nº de tabelas extraídas	35k	12B	631k
Apresenta regras de extração	não	não	sim

Tabela 3: Comparação de categorias de tabelas mencionadas em trabalhos.

Comparando as categorias de (YOSHIDA; TORISAWA, 2001) com as do presente trabalho, percebe-se que todas podem ser representadas como HORIZONTAL, VERTICAL (com e sem *labels*) e uma variação de SPLITTED, onde a tabela é partida no sentido perpendicular ao das tuplas, ao contrário da categoria apresentada no presente trabalho. Nos experimentos, eles relataram que apenas 1,7% das tabelas coletadas pertence a alguma dessas categorias partidas. Como também não encontramos quantidade relevante em nossa amostra, não incluimos em nossa taxonomia. A correspondência de suas categorias com as deste trabalho é a seguinte:

- *Type 0:* HORIZONTAL sem *labels*
- *Type 1-h:* HORIZONTAL
- *Type 1-v:* VERTICAL
- *Type 2-h:* HORIZONTAL com variação de SPLITTED
- *Type 3-h:* HORIZONTAL com variação de SPLITTED
- *Type 4-h:* HORIZONTAL com variação de SPLITTED
- *Type 2-v:* VERTICAL com variação de SPLITTED
- *Type 3-v:* VERTICAL com variação de SPLITTED
- *Type 4-v:* VERTICAL com variação de SPLITTED

Os trabalhos que propuseram taxonomias têm suas características comparadas na Tabela 3. Percebe-se que a análise mais abrangente foi feita pelo TabEx, considerando o número de tabelas da base de dados. Entretanto, no que se refere à cobertura das categorias apresentadas, o presente trabalho mostra-se superior.

5 ANÁLISE EXPERIMENTAL

Com o intuito de validar a taxonomia definida na Seção 4.1, foram extraídas tabelas reais da Web para a realização de experimentos. Para coletar o conjunto de tabelas foi feito um *crawler* como parte do TCC do aluno Marcelo Scheidt, cujo funcionamento consiste em recuperar uma URL de uma fila, acessar a página referente a ela, recolher os *links* presentes na página atual e adicioná-los à fila (CHO; GARCIA-MOLINA; WIDOM, 2001). Esse processo é repetido até que volume de páginas desejado seja atingido.

As sementes do *crawler*, i.e., páginas usadas como ponto de partida, foram sites de enciclopédias, *e-commerce*, notícias e universidades. Ao total, visitaram-se 174.927 páginas, das quais 104.261 continham tabelas. Destas, extraíram-se 631.382 tabelas HTML sem qualquer tipo de filtragem, de modo a obter uma distribuição completa na taxonomia proposta. Após descartar repetições, restaram 342.795.

5.1 AVALIAÇÃO QUALITATIVA DO WTCLASSIFIER

A fim de avaliar o resultado do WTClassifier, foi retirada uma amostra de 4.000 tabelas das obtidas através do *crawler* descrito anteriormente. Cada elemento dessa amostra foi manualmente rotulado de acordo com as categorias da taxonomia apresentada na Seção 4.1.

Para cada uma das classificações (Primária e Secundária), foi feito um processo de *stratified tenfold cross validation*, descrito na Subseção 4.2.2, a fim de estimar o erro. Como as saídas da rede neural variam entre 0 e 1, foi utilizado um *threshold* de 0,7 para determinar se a tabela em questão pertence ou não à determinada categoria.

Os resultados do teste do *WTClassifier* apresentam-se comparados com os resultados esperados nas Tabelas 4 e 5, através dos termos *true positive* (TP), *true negative* (TN), *false positive* (FP) e *false negative* (FN). Nessa terminologia, *positive* e *negative* referem-se à predição do classificador, enquanto *true* e *false* indicam se a predição corresponde ao resultado correto. Dessa forma, a coluna *Esperados* apresenta o total de tabelas que, de acordo com a categorização manual realizada, correspondem à categoria em questão; TP indica o número de elementos que sistema corretamente classificou como pertencente à categoria; TN, o total classificado corretamente como não pertencente; FP, tabelas não pertencentes que foram erroneamente classificadas como elementos da categoria; e FN, o número de tabelas da categoria não detectadas.

	Esperados	TP	TN	FP	FN
Horizontal	739	669	3048	213	70
Vertical	155	126	3770	75	29
Matrix	14	3	3976	10	11
Formatting	2862	2759	934	205	102
Navigational	230	174	3591	179	56

Tabela 4: Comparação dos resultados do WTClassifier com os esperados para a Classificação Principal.

	Esperados	TP	TN	FP	FN
Concise	88	43	613	79	45
Nested	32	9	719	29	23
Simple multivalued	350	291	323	107	59
Composed multivalued	387	304	274	119	83
Splitted	49	27	716	15	22

Tabela 5: Comparação dos resultados do WTClassifier com os esperados para a Classificação Secundária.

Para avaliar a qualidade dos resultados, foram usadas as medidas *precisão*, *revocação* e *medida F*. A primeira, precisão, consiste na razão entre a quantidade de tabelas classificadas corretamente e o total da categoria. Em outras palavras, mede a qualidade dos resultados retornados. Em função dos termos definidos no início da sessão, tem-se

$$precisao = \frac{TP}{TP+FP} \tag{5.1}$$

Por outro lado, revocação mede a quantidade de tabelas corretas recuperadas corretamente em comparação com o total disponível.

$$revocacao = \frac{TP}{TP+FN} \tag{5.2}$$

É essencial que haja um equilíbrio entre as duas medidas, e esse é o papel de medida F. Ela consiste na média harmônica de precisão e revocação.

$$medidaF = 2 \cdot \frac{precisao \cdot revocacao}{precisao + revocacao} \tag{5.3}$$

Com base nos valores das Tabelas 4 e 5 e nas Equações 5.1, 5.2 e 5.3, foram calculados precisão, revocação e medida F para as dez categorias

	Precisão	Revocação	Medida F
Horizontal	0,76	0,90	0,83
Vertical	0,63	0,81	0,71
Matrix	0,23	0,21	0,22
Formatting	0,93	0,96	0,95
Navigational	0,49	0,76	0,60

Tabela 6: Precisão, revocação e medida F para a Classificação Principal.

	Precisão	Revocação	Medida F
Concise	0,35	0,49	0,41
Nested	0,24	0,28	0,26
Simple multivalued	0,73	0,83	0,78
Composed multivalued	0,72	0,79	0,75
Splitted	0,64	0,55	0,59

Tabela 7: Precisão, revocação e medida F para a Classificação Secundária.

definidas. As Tabelas 6 e 7 apresentam os resultados numéricos considerando um *threshold* de 0,7. Já as Figuras 26 e 27 mostram precisão em função de revocação graficamente.

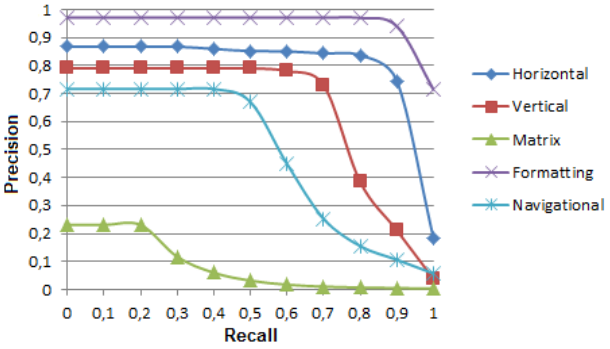


Figura 26: Gráfico de revocação vs precisão para a Classificação Principal.

Comparando os valores de medida F da Tabela 8 para as categorias em comum, percebe-se que o WTClassifier possui desempenho compatível com o reportado pelo TabEx (CRESTAN; PANTEL, 2011). Entretanto, o primeiro teve desempenho superior nas categorias HORIZONTAL, VERTICAL, FORMATTING e NAVIGATIONAL.

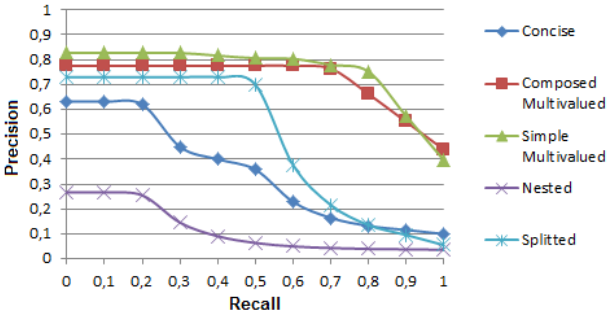


Figura 27: Gráfico de revocação vs precisão para a Classificação Secundária.

	TabEx	WTClassifier
Horizontal	0.72	0.83
Vertical	0.24	0.71
Matrix	0.30	0.22
Formatting	0.86	0.95
Navigational	0.45	0.60

Tabela 8: Medida F do TabEx e do WTClassifier.

Sobre as categorias com dados multivalorados, dois fatores contribuíram para seu bom desempenho na classificação: o número de elementos representativo no conjunto rotulado de treino e a lista de características adicional nos dados de entrada, dos quais quatro favoreciam a identificação de tabelas destas categorias.

O WTCLASSIFIER teve dificuldade em identificar corretamente as categorias com poucos exemplares na fase de treino. Percebe-se que as categorias MATRIX, CONCISE, NESTED e SPLITTED possuíam menos de 100 elementos na amostra inicial, e decorrente disso, apresentaram valores baixos na análise qualitativa: todas obtiveram medida F abaixo de 0,60.

5.2 DISTRIBUIÇÃO DE TABELAS WEB DE ACORDO COM A TAXONOMIA PROPOSTA

O conjunto com 342.795 tabelas Web passou pela Classificação Principal, onde foram divididas nas categorias de *Relational Knowledge* e *Layout*. As do primeiro grupo totalizaram 75.233 e posteriormente passaram pela Classificação Secundária. Foi definido um *threshold* de 0,7 para afirmar que

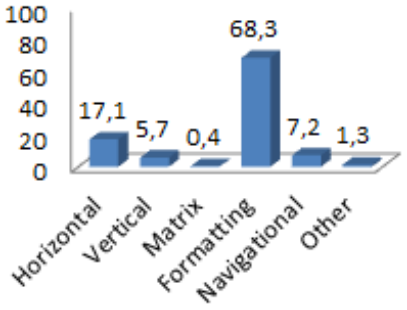


Figura 28: Distribuição de Web tables na Classificação Primária.

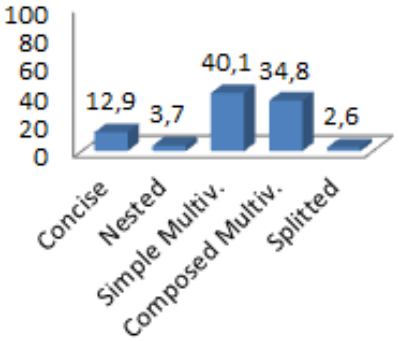


Figura 29: Distribuição de Web tables na Classificação Secundária.

uma tabela pertence à categoria.

A Figura 28 mostra a distribuição das tabelas Web de acordo com a taxonomia apresentada na Seção 4.1. Percebe-se que a estrutura mais utilizada para representar dados relacionais é HORIZONTAL (17,1%), seguida por VERTICAL (5,7%). É importante observar que as Categorias Principais são mutuamente exclusivas, i.e., uma tabela Web não pode estar em mais de uma ao mesmo tempo. Um total de 1,3% delas não atingiram o *threshold* de 0,7 para nenhuma categoria.

Já a distribuição na Classificação Secundária está representada na Figura 29. Nela, observa-se que MULTIVALUED é o tipo mais frequente, ocorrendo em 74,9% das tabelas Web com dados relacionais (40,1% INCLUSIVE e 34,8% COMPOSED). Ao contrário da Classificação Principal, as categorias não são mutuamente exclusivas.

A fim de verificar quantas tabelas Web com dados relacionais

encontravam-se formatadas de acordo com a estrutura relacional, foi contado o número de tabelas classificadas como HORIZONTAL que não se encaixavam em nenhuma categoria da Classificação Secundária. Após esse processo, verificou-se que apenas 17,75% apresentam estrutura homogênea.

6 CONCLUSÕES E TRABALHOS FUTUROS

A falta de padronização estrutural de tabelas na Web dificulta a interpretação de seus dados, mesmo que seus conteúdos sejam similares. Para maximizar a utilização de dados dessa enorme fonte, o objetivo principal do WT2RT foi realizar a classificação automática de tabelas Web heterogêneas, de modo possibilitar extração do conhecimento relacional presente nelas.

Diferentemente do estado da arte sobre tabelas Web heterogêneas, que não formaliza as categorias propostas, este trabalho apresenta definições formais de acordo com o Modelo Relacional de Codd (CODD, 1970), uma vez que se tratam de relações. Além disso, no comparativo da Tabela 2, observa-se que foram propostas três categorias que ainda não haviam sido mencionadas em trabalhos anteriores.

A partir das categorias encontradas, foi organizada uma taxonomia que engloba desde tabelas usadas para fins de formatação até estruturas semelhantes à relacional. Tendo em vista essa organização, foi implementado um classificador baseado em Redes Neurais Artificiais para, de forma automática, identificar a qual categoria cada tabela Web pertence.

Por meio de avaliação experimental, verificou-se que a qualidade do resultado de classificação automática superou o *framework TabEx* (CRESTAN; PANTEL, 2011) em 4 das 5 categorias em comum, considerando a medida F. Com relação aos trabalhos que propõem classificação de tabelas Web, esta dissertação possui o diferencial de definir algoritmos que tratam cada categoria heterogênea, de modo a trazê-las para o Modelo Relacional.

Entretanto, ainda existem desafios na classificação e no processamento de tabelas Web heterogêneas. Os principais são:

1. Realizar experimentos para validar os algoritmos de unificação estrutural propostos na Seção 4.3.
2. Medir a relevância de cada dado de entrada no WTCLASSIFIER.
3. Coletar mais casos de tabelas Web das categorias MATRIX, NESTED e CONCISE, já que essas tiveram uma baixa representatividade na amostra coletada e, consequentemente, o WTCLASSIFIER apresentou baixo desempenho na identificação delas.
4. Tratar caso de tabelas Web cujo tema não está presente em suas estruturas, e sim no texto das páginas que as contêm.
5. Estudar a possibilidade de encontrar dependências funcionais e utilizá-las para normalização das tabelas Web.

Como contribuições desta dissertação, destacam-se:

- Publicação na Escola Regional de Banco de Dados (LAUTERT; DORNELES, 2012) com experimentos preliminares sobre pré-processamento de tabelas Web (Capítulo 3).
- Publicação no periódico SIGMOD Record (LAUTERT; SCHEIDT; DORNELES, 2013) com propostas de taxonomia, formalização de categorias, WTCLASSIFIER e seus experimentos.
- Colaboração no Trabalho de Conclusão de Curso do aluno Marcelo Scheidt (UFSC), que se propôs a extrair conteúdo relacional de tabelas Web e gerar comandos SQL para inseri-lo em um banco de dados relacional.

REFERÊNCIAS BIBLIOGRÁFICAS

ABITEBOUL, S.; HULL, R.; VIANU, V. *Foundations of Databases*. [S.l.]: Addison-Wesley, 1995.

CAFARELLA, M. J. et al. Webtables: exploring the power of tables on the web. *Proc. VLDB Endow.*, VLDB Endowment, v. 1, n. 1, p. 538–549, ago. 2008. ISSN 2150-8097. <http://dx.doi.org/10.1145/1453856.1453916>.

CAFARELLA, M. J.; MADHAVAN, J.; HALEVY, A. Web-scale extraction of structured data. *SIGMOD Record*, ACM, New York, NY, USA, v. 37, n. 4, p. 55–61, mar. 2009. ISSN 0163-5808. <http://doi.acm.org/10.1145/1519103.1519112>.

CAFARELLA, M. J.; WU, E. Uncovering the relational web. In: *In under review*. [S.l.: s.n.], 2008.

CHO, J.; GARCIA-MOLINA, H.; WIDOM, J. *Crawling the Web: Discovery and Maintenance of Large-Scale Web Data*. 2001.

CODD, E. F. A relational model of data for large shared data banks. *Communications of the ACM*, ACM, v. 13, n. 6, p. 377–387, 1970.

CRESTAN, E.; PANTEL, P. Web-scale table census and classification. In: *Proceedings of the fourth ACM international conference on Web search and data mining*. New York, NY, USA: ACM, 2011. (WSDM '11), p. 545–554. ISBN 978-1-4503-0493-1. <http://doi.acm.org/10.1145/1935826.1935904>.

DEMPSTER, A. P.; LAIRD, N. M.; RUBIN, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, v. 39, n. 1, p. 1–38, 1977.

EMBLEY, D. W.; TAO, C.; LIDDLE, S. W. Automating the extraction of data from html tables with unknown structure. *Data Knowl. Eng.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 54, n. 1, p. 3–28, jul. 2005. ISSN 0169-023X. <http://dx.doi.org/10.1016/j.datak.2004.10.004>.

GATTERBAUER, W. et al. Towards domain-independent information extraction from web tables. In: *Proceedings of the 16th international conference on World Wide Web*. New York, NY, USA: ACM, 2007. (WWW '07), p. 71–80. ISBN 978-1-59593-654-7. <http://doi.acm.org/10.1145/1242572.1242583>.

LAUTERT, L. R.; DORNELES, C. F. Pré-processamento de tabelas web heterogêneas para execucão do algoritmo de juncão *merge*. In: *Escola Regional de Banco de Dados*. Curitiba, PR: [s.n.], 2012.

LAUTERT, L. R.; SCHEIDT, M. M.; DORNELES, C. F. Web table taxonomy and formalization. *SIGMOD Record*, ACM, New York, NY, USA, v. 42, n. 3, p. 28–33, set. 2013.

LIMAYE, G.; SARAWAGI, S.; CHAKRABARTI, S. Annotating and searching web tables using entities, types and relationships. *Proc. VLDB Endow.*, VLDB Endowment, v. 3, n. 1-2, p. 1338–1347, set. 2010. ISSN 2150-8097. <http://dl.acm.org/citation.cfm?id=1920841.1921005>.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, Springer New York, v. 5, n. 4, p. 115–133, dez. 1943. ISSN 0007-4985. <http://dx.doi.org/10.1007/bf02478259>.

MERGEN, S.; FREIRE, J.; HEUSER, C. A. Querying structured information sources on the web. In: *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*. New York, NY, USA: ACM, 2008. (iiWAS '08), p. 470–476. ISBN 978-1-60558-349-5. <http://doi.acm.org/10.1145/1497308.1497394>.

MERGEN, S. L. S.; FREIRE, J.; HEUSER, C. A. Indexing relations on the web. In: *Proceedings of the 13th International Conference on Extending Database Technology*. New York, NY, USA: ACM, 2010. (EDBT '10), p. 430–440. ISBN 978-1-60558-945-9. <http://doi.acm.org/10.1145/1739041.1739094>.

NAGY, G. et al. Data extraction from web tables: The devil is in the details. In: *ICDAR*. [S.l.: s.n.], 2011. p. 242–246.

RIEDMILLER, M.; BRAUN, H. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In: *IEEE INTERNATIONAL CONFERENCE ON NEURAL NETWORKS*. [S.l.: s.n.], 1993. p. 586–591.

ROJAS, R. *Neural Networks - A Systematic Introduction*. Berlin: Springer-Verlag, 1996.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. *Sistema de Banco de Dados*. [S.l.: s.n.], 2006. 25–48 p.

SILVA, A. S. da et al. Labeling data extracted from the web. In: *Proceedings of the 6th International Conference on Ontologies,*

DataBases, and Applications of Semantics. Springer, 2007. p. 1099–1116. <http://dx.doi.org/10.1007/978-3-540-76848-7_72>.

VENETIS, P. et al. Recovering semantics of tables on the web. *Proc. VLDB Endow.*, VLDB Endowment, v. 4, n. 9, p. 528–538, jun. 2011. ISSN 2150-8097. <http://dl.acm.org/citation.cfm?id=2002938.2002939>.

WANG, Y.; HU, J. Detecting tables in html documents. In: *Proceedings of the 5th International Workshop on Document Analysis Systems V*. London, UK, UK: Springer-Verlag, 2002. (DAS '02), p. 249–260. ISBN 3-540-44068-2. <http://dl.acm.org/citation.cfm?id=647798.736657>.

WANG, Y.; HU, J. A machine learning based approach for table detection on the web. In: *Proceedings of the 11th international conference on World Wide Web*. New York, NY, USA: ACM, 2002. (WWW '02), p. 242–250. ISBN 1-58113-449-5. <http://doi.acm.org/10.1145/511446.511478>.

WITTEN, I. H.; FRANK, E. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005. ISBN 0120884070.

YOSHIDA, M.; TORISAWA, K. A method to integrate tables of the world wide web. In: *In Proceedings of the International Workshop on Web Document Analysis (WDA 2001)*. [S.l.: s.n.], 2001. p. 31–34.