IComp/UFAM

APRENDENDO A SEGMENTAR PÁGINAS WEB

Caio Moura Daoud

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática, Instituto de Computação - IComp, da Universidade Federal do Amazonas, como parte dos requisitos necessários à obtenção do título de Mestre em Informática.

Orientador: David Braga Fernandes de Oliveira

Março de 2013

Manaus - AM

APRENDENDO A SEGMENTAR PÁGINAS WEB

Caio Moura Daoud

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO DO INSTITUTO DE COMPUTAÇÃO DA UNIVERSIDADE FEDERAL DO AMAZONAS COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM INFORMÁTICA.

Aprovado por:

Prof. David Braga Fernandes de Oliveira, D.Sc.

Prof. Eulanda Miranda dos Santos, D.Sc.

Prof. Thierson Couto, D.Sc.

MARÇO DE 2013

MANAUS, AM – BRASIL

Resumo

Diferente dos documentos tradicionais, as páginas Web são compostas por diferentes segmentos ou blocos, cada qual desempenhando uma função específica dentro de cada página.Trabalhos recentes da literatura têm demonstrado que informações sobre esses segmentos podem ser úteis para melhorar os resultados de inúmeras tarefas das áreas de recuperação de informação e mineração de dados. Por esse motivo, existem muitos trabalhos científicos propondo diferentes métodos de segmentação de páginas Web. De uma forma geral, os métodos de segmentação encontrados na literatura utilizam apenas evidências da própria página a ser segmentada. No entanto, partindo da observação de que as páginas de um mesmo site tendem a possuir layouts bastante similares, apresentamos neste trabalho uma abordagem baseada em aprendizagem de máquina que explora evidências globais dos Web sites. Nosso método, que adota Support Vector Machines para o processo de aprendizagem, e usa a estrutura SOM (Site Object Model) para agregar informações de todas as páginas de um mesmo Web site, apresentou bons resultados quando comparado com uma abordagem de segmentação manual, e quando comparado com uma recente abordagem da literatura.

PALAVRAS-CHAVE: Segmentação de páginas Web, Aprendizagem de máquina, Árvore SOM

Abstract

Unlike traditional documents, Web pages are composed of different segments or blocks, each block has specific functions in each page. Recent work in the literature has shown that information on these segments may be useful to improve the results of numerous tasks in information retrieval and data mining areas. For this reason, there are many scientific works proposing different methods for Web pages segmentation. Generally speaking, the targeting methods found in the literature only use evidences of the page to be segmented. However, based on the observation that the pages of a site tend to have very similar layouts, we present a strategy based on machine learning that explores overall evidences of Web sites. Our method, which adopts Support Vector Machines for the learning process, and use the SOM structure (Site Object Model) to aggregate information from all pages of a Web site, achieved good results when compared a manual segmentation approach, and with a recent approach in the literature.

KEY-WORDS:: Web pages segmentation, machine learning, SOMtree

Sumário

Lista de Figuras

Capítulo 1

Introdução

Páginas Web, em sua maioria, são formadas por regiões que compõem sua estrutura, tais como: título, conteúdo principal, menus e propagandas. Trabalhos recentes demonstraram que o uso desses elementos estruturais por sistemas de recuperação de informação pode resultar em melhorias significativas na qualidade do ranking gerado [14, 11]. Esses elementos também podem ser aplicados na solução de problemas de reestruturação de páginas para dispositivos com tela reduzida. Na área de recuperação de informação, as informações dessas regiões são usadas para identificar a real importância de cada região da página. Nesse sentido, algumas pesquisas vêm sendo realizadas com o objetivo de identificar e atribuir pesos às regiões das páginas, com o intuito de melhor representar o conteúdo das páginas no processamento da busca. Nos problemas de reestruturação de páginas para dispositivos com tela reduzida, os pesos aplicados a cada região são utilizados em um processo de identificação das regiões que devem ser exibidas no novo formato [17]. Nesse contexto, métodos de segmentação de páginas vêm sendo utilizados para identificar os elementos estruturais presentes nas páginas.

Em geral, as abordagens tradicionais de segmentação encontradas na literatura necessitam de intervenção humana ou possuem funções e limiares definidos por meio de abordagens empíricas. Neste trabalho, apresentamos um método de segmentação automática de páginas Web que não requer intervenção humana e que aplica técnicas de aprendizagem de máquina para gerar as funções e os limiares que são aplicados no processo de segmentação.

Outra característica das abordagens tradicionais de segmentação é a utilização de evidências extraídas unicamente da página que será segmentada. Esses métodos desconsi-

deram que em um site há várias páginas que possuem o mesmo layout (estrutura), o que poderia fornecer evidências mais relevantes para o processo de segmentação. Em [14], os autores demonstram que o uso de informações de todo o site no processo de segmentação automática auxilia na obtenção resultados mais próximos de uma segmentação manual. Nosso método faz uso de uma estrutura em árvore chamada SOM (Site Object Model), proposto em [14], que é capaz de representar o conteúdo e a estrutura de árvores de todas as páginas de um site. Essa estrutura se baseia em uma estratégia de alinhamento das árvores DOM (Document Object Model) de um site. A utilização da árvore SOM no processo de segmentação permite identificar os blocos presentes em todas as páginas de um Web site, agrupando blocos equivalentes em classes de blocos.

Em nosso trabalho demonstramos que o uso de técnicas de aprendizagem de máquina no processo de segmentação automática, baseado na árvore SOM, produz resultados superiores aos reportados em nosso baseline [13]. A avaliação foi feita por meio de métricas que calculam a similaridade entre a segmentação resultante dos métodos a serem avaliados com a segmentação feita manualmente. Foram usadas coleções de páginas previamente segmentadas por especialistas como entrada para os métodos de aprendizagem de máquina aplicados na geração de estatísticas para encontrar blocos em páginas ainda não segmentadas.

Consideramos como contribuição deste trabalho o aperfeiçoamento do processo de segmentação proposto em [13], ao substituirmos funções *ad hoc* que utilizam limiares definidos empiricamente por funções mais robustas e mais estáveis geradas automaticamente a partir de exemplos extraídos de páginas Web utilizadas como referência.

1.1 Organização do Trabalho

Este trabalho está estruturado da seguinte forma. No Capítulo 2, são apresentados os conceitos básicos para o entendimento deste trabalho e os alguns métodos de segmentação de páginas encontrados na literatura. O Capítulo 3, apresenta o método de segmentação utilizado como baseline. No Capítulo 3.3, é descrito o processo de aprendizagem de máquina utilizado neste trabalho. No Capítulo 4, apresentamos os experimentos e os resultados alcançados. Por fim, no Capítulo 5, é feita conclusão e são apresentadas propostas para trabalhos futuros.

Capítulo 2

Conceitos Básicos

Neste capítulo, são apresentados os fundamentos relacionados ao problema de segmentação de páginas. A Seção 2.1 apresenta os conceitos de blocos em páginas Web, assim como o conceito de árvore DOM. A Seção 2.2 aborda os conceitos de aprendizagem de máquina que são aplicados neste trabalho.

2.1 Estrutura em Bloco das Páginas Web

Ao visualizar páginas Web através de um browser, é possível perceber que o conteúdo dessas páginas é normalmente composto por diferentes regiões, tais como o conteúdo principal, os menus, as propagandas, etc. Também é possível perceber que cada uma dessas regiões desempenha uma função específica dentro de sua respectiva página. No trabalho realizado em [13], essas regiões são chamadas de *segmentos* ou blocos, neste trabalho adotamos o termo *bloco* para nos referir a essas regiões. Um bloco é uma região lógica auto-contida dentro de uma página Web que: (i) não está aninhada com outros blocos e (ii) é representada por um par (l, c), onde l é o rótulo (*label*) do bloco, representado pelo caminho entre a raiz e o bloco na árvore DOM, e c é a porção de texto que o bloco possui.

Para um ser humano, a identificação dos blocos de uma página é feita através de sua percepção visual. Por exemplo, ao observar a Figura 2.1, um usuário não teria problema em identificar que o título, o menu e o conteúdo principal formam três blocos de conteúdo independentes. Da mesma forma, o usuário é capaz de perceber facilmente que cada um desses blocos é composto por fragmentos ainda menores, tal como, em nosso exemplo, o

Figura 2.1: Exemplos de blocos em um trecho de página do Web site IG

menu é constituído por diferentes links.

2.1.1 Árvore DOM

Para um computador, uma página é representada por um conjunto de elementos de uma estrutura hierárquica chamada DOM (*"Document Object Model"*), que define a lógica e a organização do conteúdo das páginas Web. Os nós internos da árvore DOM correspondem às tags do código HTML da página e os nós folha correspondem ao conteúdo da página, como texto, imagens e links.

Na Figura 2.2 temos um exemplo de como uma página HTML é mapeada para uma árvore DOM. Podemos observar que a relação pai-filho entre os nós da árvore corresponde à hierarquia das tags presentes no HTML. Na árvore DOM cada nó possui as seguintes informações: (i) **nome da tag:** identificador do nó; (ii) **atributos:** lista com os atributos da tag, sendo cada atributo identificado por uma tupla (*nome*, *valor*); e **conteúdo:** conteúdo do nó, quando houver texto.

Enquanto uma página é representada em um computador por uma árvore DOM, um bloco é representado por uma sub-árvore desta estrutura hierárquica, possuindo portanto um nó raiz e um conjunto de nós internos. Cada bloco possui um *label*, formado por uma *string* contendo o nome das tags presentes no caminho entre o bloco e a raiz da árvore

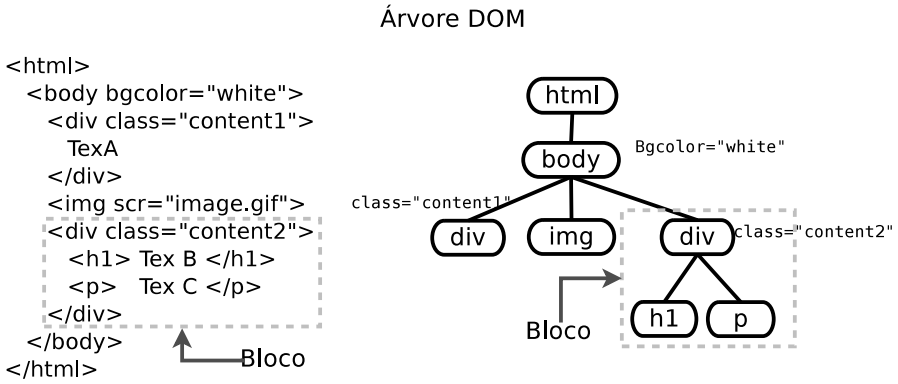

Figura 2.2: Árvore DOM

DOM. Nessa string, o nome das tags são separadas por barras. Por exemplo, o bloco em destaque da Figura 2.2 é constituído pelo nó *DIV*, à direita da árvore, seguido de seus filhos $H1$ e P. Para esse exemplo, o *label* deste bloco é $DIV/BODY/HTML$. Neste caso, o trabalho de um algoritmo de segmentação de páginas é identificar que nós desta sub-árvore fazem parte de um único bloco.

2.1.2 Classes de bloco

O conceito de classes de blocos surge ao considerarmos a similaridade entre as estruturas das páginas de um Web site. Cada site é representado por um conjunto de páginas $S = (p_1, p_2, ..., p_n)$, sendo cada página representada por um conjunto de blocos $p = (b_1, b_2, ..., b_m)$. O número m de blocos de uma página varia conforme a estrutura interna desta página. Partindo desse princípio e considerando que em sua maioria as páginas de um Web site possuem estruturas similares, uma classe C de blocos é definida como um conjunto de blocos que pertencem à páginas distintas e que possuem o mesmo *label* na árvore DOM de suas respectivas páginas. Esses conceitos podem ser definidos formalmente como:

$$C = (b_1, b_2, ..., b_{nc})$$
$$b_1 = (label_c, texto_1)$$
$$b_2 = (label_c, texto_2)$$
$$...$$
$$b_{nc} = (label_c, texto_{n_c})$$

Para exemplificar este conceito vamos utilizar a Figura 2.3 que é composta por trechos de duas páginas de um mesmo Web site e que possuem estrutura similar. Observe que os blocos *Título* das páginas dispostas na figura pertencem a uma mesma classe. Tal fato pode ser constatado observando que cada um desses blocos está situado na mesma posição dentro de suas respectivas páginas e conseqüentemente possuem o mesmo *label* dentro de suas respectivas árvores DOM. Além do bloco *Título*, as duas páginas da Figura 2.3 possuem outros blocos em comum. Por exemplo, ambas as páginas possuem um bloco *Menu*, que também pertencem a uma mesma classe por possuírem o mesmo *label*. Outra característica que pode ser observada neste exemplo é que os blocos de uma mesma classe tendem a exercer uma mesma função dentro de suas respectivas páginas.

Neste trabalho utilizamos para o processo de segmentação automática uma estrutura que combina todas as páginas de um Web site em uma única árvore. O processo sobre essa estrutura leva a identificação das classes de blocos do Web site. A partir dessas classes, chegamos a estrutura em blocos de cada página. Outro fator importante para a identificação dessas classes de blocos existentes nos Web sites é a existência de trabalhos que demonstram que o uso dessa informação pode ser útil para melhoria do *ranking* em sistemas de recuperação de informação.

Figura 2.3: Exemplo de páginas com a mesma estrutura extraídas do Web site IG

2.2 Aprendizagem de Máquina

Durante o processo de segmentação automática desenvolvido neste trabalho são realizadas sucessivas tomadas de decisão para identificar os nós da árvore DOM que representam os

blocos da página Web. É neste processo de tomada de decisão ou de classificar elementos que aplicamos técnicas de aprendizagem de máquina. Nesta subseção, apresentamos uma breve introdução a essas técnicas, assim como ao método de classificação utilizado em neste trabalho.

Na área de Aprendizagem de Máquina (AM), do inglês *Machine Learning* [30], são desenvolvidas técnicas para que um sistema computacional possa tomar decisões de forma automática. Esse processo de tomada de decisões é feito com base no princípio de inferência, procurando estabelecer conclusões genéricas a partir de um conjunto particular de exemplos. Ou seja, com base em exemplos, um método de aprendizagem pode definir limiares ou regras genéricas formando um modelo do conhecimento. Essas técnicas são conhecidas como aprendizagem por indução e podem ser diferenciadas em três tipos: aprendizagem supervisionada, não supervisionada e aprendizagem por reforço [26].

O tipo de aprendizagem abordado neste trabalho é o supervisionado, usado em problemas de classificação ou previsão. Neste tipo de aprendizagem são necessários alguns exemplos de elementos já classificados ou preditos, consistindo de uma base de dados com exemplos na forma (X_i, Y_i), em que X_i representa os dados de um exemplo e Y_i representa o rótulo ou a saída desejada para os dados do exemplo. A partir do conjunto de exemplos, os algoritmos de aprendizagem procuram descobrir que características destes elementos pré-classificados ou preditos determinam a classe a qual pertencem. Feito isso, é gerado um classificador, também denominado modelo, preditor ou hipótese que é utilizado na classificação de novas entradas não apresentadas previamente. O classificador também pode ser visto como uma função onde dado uma entrada X_i é gerada uma saída (rótulo) Y_i.

2.2.1 Nomeclaturas básicas

As nomenclaturas descritas nesta seção estão relacionadas ao conjunto de termos que ajudam a compreender a montagem de uma base de dados para métodos de AM. Estes termos são: Evidência, Instância, Valor e Classe.

No aprendizado supervisionado é fornecido ao sistema de aprendizado um conjunto de exemplos denominados instâncias $I = \{I_1, I_2, ..., I_n\}$. Para cada instância é associado um rótulo que define a classe a qual ela pertence. Pode-se dizer que cada instância $I_i \in I$ é uma tupla $I_i = (\overrightarrow{x}_i, y_i)$, onde $\overrightarrow{x}_i$ é um vetor de valores que representam as evidências

da instância e y_i é a classe da instância.

Uma evidência descreve uma característica ou aspecto que uma instância pode possuir. Tomando como exemplo, em uma tabela de uma base de dados, cada evidência é habitualmente uma coluna da tabela; Uma instância é um conjunto de valores, um para cada evidência. Tomando como exemplo, em uma tabela de uma base de dados, uma instância é habitualmente uma linha da tabela; Um valor é uma característica que uma evidência pode assumir para uma determinada amostra. Tomando como exemplo, em uma tabela de uma base de dados, cada valor é o dado contido em cada célula ou campo da tabela; As classes são os domínios nos quais as amostras podem ser agrupadas. Em uma base pré-classificada, cada instância pertence a uma classe já estabelecida. Tais bases pré-classificadas são utilizadas por técnicas de aprendizagem de máquina para determinar como as novas instâncias serão classificadas.

2.2.2 Suporte vector machines

Neste trabalho utilizamos para o processo de classificação as Máquinas de Vetores de Suporte (SVM, do Inglês *Support Vector Machines*) que constituem uma técnica que vem recebendo crescente atenção da comunidade de AM. Exemplos de sucesso com aplicação dessas técnicas podem ser encontrados em diversos domínios, como na categorização de textos [19], na análise de imagens [20, 25] e em Bio-informática [15].

Desenvolvida por Vapnik [28], SVMs são baseadas na teoria de aprendizado estatístico. Essa teoria estabelece uma série de princípios para obtenção de classificadores com boa capacidade de prever corretamente a classe de novos dados do mesmo domínio em que o aprendizado ocorreu.

A figura 2.4 mostra um espaço bidimensional com duas classes representadas por *circulo aberto* e *circulo fechado*. O funcionamento de SVM pode ser descrito da seguinte forma: dadas as duas classes e o conjunto de instâncias de treinamento (os círculos na figura), SVM constrói um hiperplano que divide o espaço de características em duas regiões, maximizando a margem de separação entre as mesmas como demonstrado na figura em questão. Esse hiperplano é conhecido como hiperplano de separação ótima. Para teste do hiperplano, amostras desconhecidas são mapeadas para esse mesmo espaço, e atribuídas a uma das classes [3].

As retas pontilhadas $H1$ e $H2$, paralelas ao hiperplano, constituem o par de hiperpla-

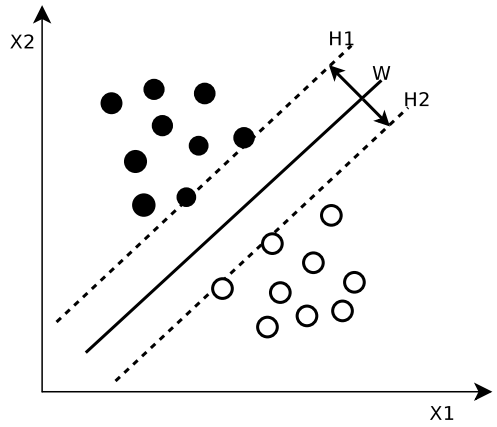

Figura 2.4: Espaço bidimensional com duas classes separadas por um hiperplano de separação ótima.

nos que geram a margem máxima pela minimização do vetor. Os pontos que estão nos hiperplanos $H1$ e $H2$ são chamados vetores de suporte. Caso esses pontos sejam removidos, a solução encontrada é alterada. Além disso, em fase de uso do SVM, apenas os vetores de suporte são necessários para que dados desconhecidos sejam classificados.

Diferente do exemplo da Figura 2.4, a maioria dos problemas reais não são linearmente separáveis [3]. SVM trata esses problemas com o uso de diferentes funções de kernel que possibilitam a construção de SVM com diferentes tipos de superfícies de decisão não-linear no espaço de entrada.

Com o uso de funções kernel, as instâncias são inicialmente mapeadas para um espaço de características de maior dimensão que o espaço de características original. Permitindo dessa forma, a classificação em espaços não linearmente separáveis. A Figura 2.5 mostra o processo de transformação de um domínio não linearmente separável, em um problema linearmente separável através do aumento da dimensão, consequência do mapeamento feito por uma função kernel $F(x)$.

Podemos observar no quadro 1 da Figura 2.5 que não existe um separador linear para esse problema. No entanto, se mapearmos cada vetor de entrada x para um novo vetor de características $F(x)$, é possível construir o hiperplano de separação no quadro 2 da Figura 2.5. Então, o que uma máquina de vetor suporte faz é mapear os dados para um espaço de dimensão suficientemente alta. Dentre as funções kernel mais usadas destacam-se: Polinômios, Funções de Base Radial (RBF) ou Gaussiana e Sigmóide, definindo diferentes máquinas de aprendizagem.

Em [23], os autores concluem que as SVMs são robustas diante de dados de grande

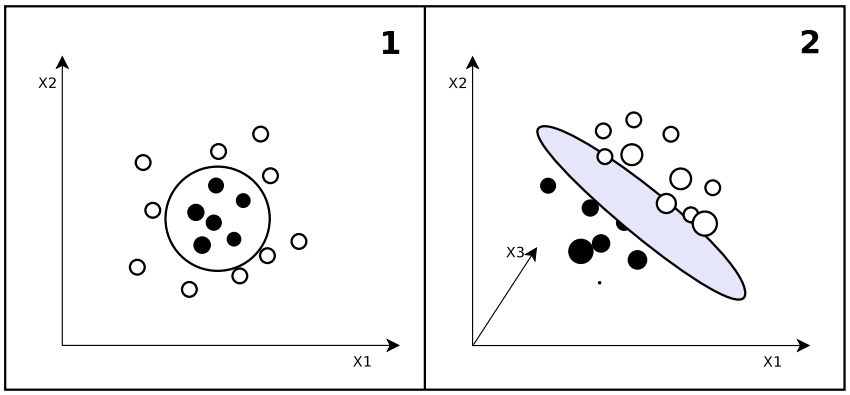

Figura 2.5: Conversão de um espaço bidimensional não linearmente separável a um espaço em que os vetores podem ser linearmente separados.

dimensão, sobre os quais outras técnicas de aprendizado comumente obtém classificadores muito específicos ou muito genéricos. Além disso, reportam que o uso de funções Kernel torna o algoritmo eficiente, pois permite a construção de simples hiperplanos em um espaço de alta dimensão de forma tratável do ponto de vista computacional [5].

Entre as principais limitações de SVM pode-se destacar a sua sensibilidade à escolha de valores de parâmetros e a dificuldade de interpretação do modelo gerado por essa técnica, problemas que têm sido abordados em diversos trabalhos como [9, 12, 18] e [16, 6] respectivamente.

2.2.3 LibSVM (*A Library to Support Vector Machine*)

Para implementação das técnicas de aprendizagem de máquina com SVM utilizamos a biblioteca LibSVM. LibSVM é uma biblioteca que utiliza SVM para classificação e regressão de padrões [8]. Há versões da biblioteca para vários sistemas operacionais e pode ser encontrada em várias linguagens como C++, C# e Java. Além de ser gratuita e de código aberto, ela também permite conexão com outros aplicativos como Matlab e Ruby.

LibSVM possui ferramentas como, SVMToy, SVMTrain, SVMPredict e SVMScale que possibilitam uma fácil execução de experimentos por parte do usuário. A SVMToy é um applet simples que demonstra classificação e regressão utilizando SVMs. A SVMTrain permite o treinamento dos dados. Partindo de um arquivo contendo os dados em um formato específico, o SMVTrain treina e gera um arquivo do tipo MODEL. Tal arquivo é utilizado para classificação. A SVMPredict utiliza duas entradas, um arquivo MODEL e um arquivo contendo dados não apresentados durante o treinamento, essa ferramenta

classifica os dados gerando um arquivo de saída. A SVMScale é utilizada para normalizar dados, sendo importante para eliminar possíveis discrepâncias entre os valores dos dados.

2.3 Avaliação de Segmentação

Utilizamos duas métricas diferentes para avaliar a qualidade do método de segmentação proposto neste trabalho: (i) Índice Ajustado De Rande e (ii) Informação Mútua Normalizada. Tais métricas são frequentemente utilizadas para avaliação de algoritmos de *clustering* em que é conhecido um agrupamento a *priori* e se pretende aferir a capacidade que diversas metodologia têm de recuperar essa estrutura.

Chakrabarti et al. [7], Kohlschutter et al. [21] assim como Fernandes et al. [13] fizeram uso dessas métricas para avaliar a qualidade de seus respectivos métodos de segmentação de páginas Web. Nestes artigos foram comparados o resultado da segmentação automática com a segmentação realizada de forma manual.

2.3.1 Índice ajustado de Rand

O índice ajustado de Rand, ou AdjRAND (*Adjusted Rand Index*) é uma medida bastante utilizada para avaliação de métodos de *clustering*. Utiliza critérios de validade externos, isso significa que AdjRAND avalia o grau de correspondência entre a estrutura de grupos gerada pelo sistema e uma solução esperada para os mesmos dados [29]. A avaliação é realizada então, a partir da comparação da segmentação automática em ρ com a segmentação manual de referência ρ' sobre uma mesma página. O valor de AdjRAND pode variar entre 0 e 1. Sendo 1 quando as segmentações ρ e ρ' são idênticas e 0 quando as segmentações ρ e ρ' são completamente diferentes. Seu cálculo se dá através da seguinte Equação 2.1.

$$AdjRand(\rho,\rho') = \frac{(\frac{n}{2})(tp+tn) - [(tp+fp)(tp+fn)+(fn+tn)(fp+tn)]}{(\frac{n}{2})^2 - [(tp+fp)(tp+fn)+(fn+tn)(fp+tn)]} \quad (2.1)$$

Uma decisão é positiva verdadeira (tp) quando duas palavras que fazem parte de um mesmo bloco são corretamente classificadas. Uma decisão é negativa verdadeira (tn) quando palavras que não fazem parte de um mesmo bloco não são atribuídas correta-

mente a blocos distintos. Quando palavras que não fazem parte de um mesmo bloco são erroneamente agrupadas em um mesmo bloco, tem-se uma decisão positiva falsa (fp). Quando temos palavras que deveriam fazer parte de um mesmo bloco e são atribuídas a blocos diferentes, tem-se então uma decisão negativa false (fn).

2.3.2 Informação mútua normalizada

A MI (Mutual Information) visa quantificar as informações compartilhadas entre dois particionamentos.

Dada uma variável x e outra y que representam respectivamente a segmentação em bloco por um método automático e por um método manual, a MI dessas duas variáveis é expressa por $I(x,y)$. No entanto, a MI não possui um limite superior, o que inviabiliza sua utilização para comparação entre diferentes métodos. Por esse motivo, uma normalização para essa métrica foi proposta por [27], chamada de Informação Mútua Normalizada, ou NMI (Normalized Mutual Information). A NMI mede a dependência mútua de duas soluções através da relação entre suas entropias. É portanto a informação mútua entre duas segmentações, normalizadas pela média de suas entropias. Sendo $H(x)$ a entropia de x, o cálculo do NMI se da através da seguinte Equação 2.1.

$$NMI(x,y) = \frac{I(x,y)}{\sqrt{H(x)H(Y)}} \tag{2.2}$$

2.4 Trabalhos Relacionados

Nesta seção apresentamos uma visão geral de algumas técnicas de segmentação encontradas na literatura. Até onde sabemos, o primeiro método de segmentação foi proposto por Cai et al [10]. Esse método, que levou o nome de VIPS (Vision Based Page Segmentation), procura segmentar as páginas através da simulação de percepção visual humana. No entanto, o VIPS exige um considerável esforço humano para definir manualmente parâmetros para cada página a ser segmentada. Apesar de gerar bons resultados, o fato de não se tratar de um método automático de segmentação, limita de forma considerável a sua aplicabilidade.

Hattori et al [17] propõem um método de segmentação baseado na distância entre elementos da árvore DOM. Essa distância é calculada com base no número de *tags* entre

dois elementos, e na profundidade desses elementos no código HTML. Combinando esse cálculo com uma análise do *layout* da página, Hattori et al [17] propõem um método híbrido de segmentação. Essa distância foi adotada como uma das evidências para o processo de aprendizagem de máquina neste trabalho.

Chakrabarti et al [7] propõem uma solução para o problema de segmentação de páginas usando algoritmos de agrupamento em grafos ponderados. Nesse método de segmentação, os pesos das arestas do grafo capturam a probabilidade de duas regiões adjacentes de uma mesma página fazerem parte de um mesmo bloco da página. Essa estratégia de avaliar elementos para determinar se eles fazem parte ou não de um mesmo bloco é adotada neste trabalho através de técnicas de aprendizagem de máquina.

Em Kohlschutter et al [21], o problema de segmentação de páginas Web é examinado a partir de uma perspectiva textual. O pressuposto fundamental neste trabalho é que a densidade textual, ou seja, o número de termos de uma dada região de uma página é um recurso valioso para as decisões de segmentação. Partindo desse pressuposto, os autores apresentam um algoritmo capaz de identificar se duas regiões de uma mesma página fazem parte de um mesmo bloco usando métricas de densidade do texto. Os autores mostram que seu método supera aquele proposto em [7]. Essa densidade de texto nos elementos é uma das evidências que adotamos para o processo de aprendizagem de máquina adotado neste trabalho.

Fernandes et al. [13] propõem um método automático de segmentação que considera não apenas informações de uma dada página a ser segmentada, mas considera evidências de todo o site dessa página. Os bons resultados desse método advém do fato de que muitas páginas de um mesmo site possuem estruturas e `layout` semelhantes. Após combinar as páginas em uma única estrutura denominada Árvore SOM, o método proposto era capaz de identificar os blocos com base na distância entre os elementos da SOM e no número de páginas do site que possuem tais elementos. Experimentos apresentados em [13] demonstraram resultados superiores aos reportados por [21]. Além disso, por se tratar de um trabalho mais recente, e por ser totalmente automático, o método de [13] será adotado como baseline para os experimentos deste trabalho. Em nosso método também utilizamos uma estrutura orientada a site para decidir quando regiões de uma mesma página fazem parte de um mesmo bloco ou não, no entanto diferente do baseline onde essas decisões eram tomadas através de limiares definidos empiricamente, neste trabalho aplicamos téc-

nicas de aprendizagem de máquina para gerar classificadores e limiares automaticamente.

Alcic et al. [1] é um dos mais recentes trabalhos sobre segmentação encontrados na literatura. Os autores reduziram o problema de segmentação de páginas a um problema de agrupamento, utilizando três funções de distância entre conteúdos propostas por eles. Essas distâncias são: (i) Distância baseada em DOM, onde são atribuídos pesos a cada nó da árvore DOM, os nós são ordenados e a distância entre dois nós é dada pela soma dos pesos entre eles na lista ordenada.(ii) Distância geométrica, em que o conteúdo Web é representado por dois pontos que formam a diagonal de um retângulo (à direita em cima, esquerda à baixo) que corresponde ao espaço deste conteúdo em uma página renderizada, sendo a distância entre dois elementos a menor distância entre os retângulos que os representam. (iii) Distância semântica, onde através de formulas baseada em IDF (inverse document frequency) que gera um valor entre 0 e 1. A distância semântica expressa a relação entre o par de conteúdo.

Capítulo 3

Segmentação Automática

A estratégia de segmentação adotada neste trabalho é uma adaptação da estratégia proposta em Fernandes et al. [13], que é composta por três passos. No primeiro passo é feito um pré-processamento de cada árvore DOM, resultando em uma versão inicial da segmentação de cada página. O segundo passo é a construção de uma estrutura em árvore chamada SOM (Site Object Model), a partir das páginas já pré-processadas. No terceiro e último passo, os blocos são finalmente identificados através da árvore SOM.

A diferença entre a estratégia adotada neste trabalho e a estratégia de Fernandes et al. [13]. dá-se no terceiro passo descrito acima, que é o passo mais importante em ambas os trabalhos. Fernandes et al. [13] propos uma abordagem empírica para a identificação dos blocos a partir da SOM, enquanto neste trabalho adotamos uma abordagem baseada em aprendizagem de máquina.

Neste Capítulo, faremos uma breve descrição dos dois primeiros passos do processo de segmentação adotados por ambos os trabalhos. Adicionalmente, descreveremos o terceiro passo do processo de segmentação conforme proposto por [13]. A abordagem de aprendizado de máquina proposta neste trabalho será descrita no capítulo seguinte.

3.1 Passo 1: Pré-processamento da Árvore DOM

Neste passo do processo de segmentação, cada árvore DOM é submetida a um processo de poda, procurando eliminar ruídos que poderiam atrapalhar o processo de segmentação desempenhado na terceira etapa do método. Esse passo é desenvolvido através de duas etapas: (i) poda de regiões aninhadas com conteúdos textuais, e (ii) poda de regiões com

estrutura regular.

3.1.1 Poda de regiões aninhadas a conteúdos textuais

Em geral, o conteúdo textual das páginas estão dispóstos nas folhas de suas respectivas árvores DOM. Quando há conteúdo textual em um nó interno, os filhos deste nó estão relacioandos a esse conteúdo e portanto são considerados elementos pertencentes a um mesmo bloco. Neste sentido, são realizados ajustes para que o conteúdo textual esteja disposto sempre nas folhas das árvores DOM. Nessa etapa do pré-processamento procuramos identificar e remover nós que possuam algum ancestral com conteúdo textual. Para exemplificar esse processo, considere a árvore DOM da Figura 3.1. Observe que o nó <*p*> possui conteúdo textual e é um nó interno, já que possui dois nós <*strong*> descendentes à ele. Nesse caso, removemos os dois nós <*strong*> da árvore DOM, e aglutinamos o conteúdo desses nós com o conteúdo de <*p*>. Desta forma, após o pré-processamento mostrado em nosso exemplo, as tags <*strong*> deixam de existir e todo o conteúdo desses dois nós passa a ser associado à tag <*p*>.

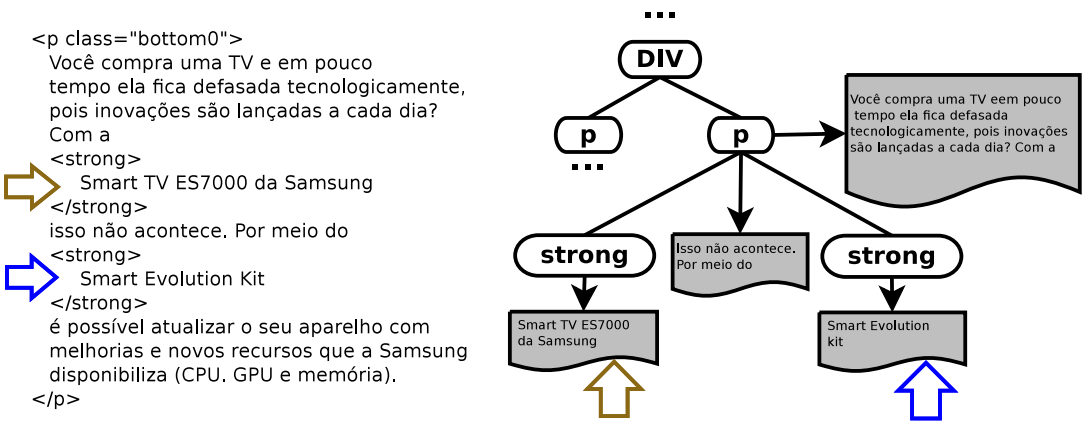

Figura 3.1: Trecho de uma página HTML com região aninhada à conteúdos textuais

3.1.2 Poda de região com estrutura regular

Na árvore DOM de uma página web típica, é comum encontrarmos regiões cujos nós estão dispostos de uma forma bastante regular. Um exemplo de região regular pode ser visto no bloco Menu da Figura 2.1, cujos links estão ordenados seguindo um mesmo padrão estrutural. A Figura 3.2(a) apresenta um trecho de uma árvore DOM responsável por apresentar outro exemplo de menu. Observe que o conjunto de tags que separam os

ítens desse menu são sempre os mesmos, de forma a caracterizar uma região de estrutura regular dentro da página. Essas regiões normalmente são formadas por conjuntos de ítens relacionados, como conjuntos de links (formando um menu), parágrafos (formando um texto) e outros. Como o conteúdo desses ítens está relacionado, dizemos que eles formam um único bloco dentro de sua respectiva página.

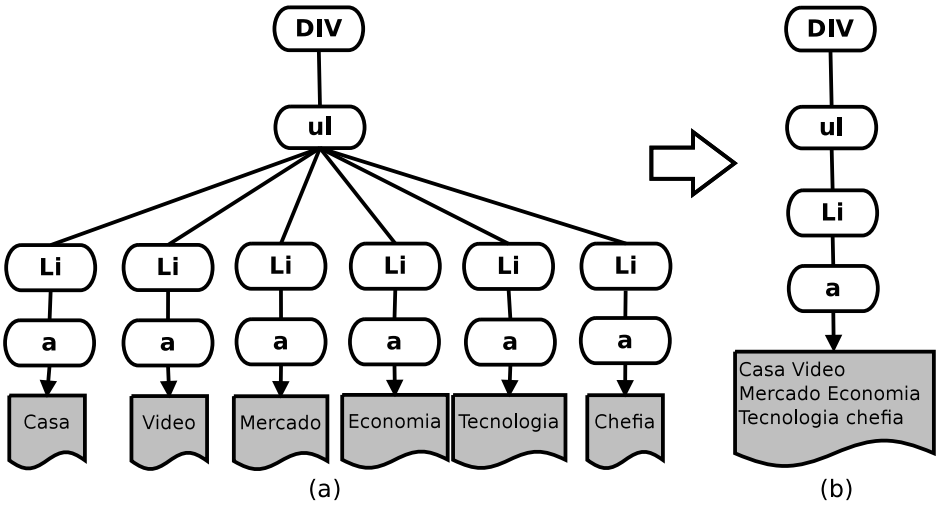

Figura 3.2: (a) Nós de uma árvore DOM com estrutura regular. (b) Resultado da poda sobre a estrutura regular da árvore (a)

Esta etapa do pré-processamento é responsável por aglutinar todas as tags das regiões regulares em uma única tag. A Figura 3.1(b) mostra a sub-árvore resultante dessa aglutinação. Observe que todo o conteúdo textual das tags removidas foram combinadas em uma única tag.

O problema de identificar regiões regulares em árvores DOM é bastante recorrente na área de extração de dados [22, 24, 4]. Em muitas ocasiões os dados a serem extraídos das páginas estão dispostos de forma regular, uma vez que padrões repetitivos podem ser encontrados nas páginas web quando múltiplos registros são agrupados. Para exemplificar, em [2] os autores buscaram identificar regiões com estrutura regular nas páginas Web a partir de informações sobre a disposição visual dos elementos na página, com o propósito de encontrar e extrair dados em documentos *Web*.

3.1.3 Passo 2: Construção da Árvore SOM

Uma vez finalizado o pré-processamento nas árvores DOM de um site, estas são combinadas em uma única estrutura denominada árvore SOM. A construção dessa estrutura

parte da observação de que, em um mesmo site, é comum haver conjuntos de páginas com estruturas similares. Por exemplo, a Figura 2.3 mostra um par de páginas com conteúdos diferentes, mas que possuem exatamente os mesmos blocos. Em um conjunto de páginas com os mesmos blocos, cada página é uma evidência de como este conjunto de páginas deve ser segmentado. Baseado nesta idéia, Fernandes et al. [13] propuseram um método de segmentação utilizando uma estrutura denominada Árvore SOM, uma variação do SST (*Site Style Tree*), proposto em [31], capaz de agregar informações de todas as páginas de um mesmo site.

Uma árvore SOM é criada através da combinação das árvores DOM das páginas de um site. A Figura 3.3 mostra um exemplo de árvore SOM gerada a partir das páginas ρ_1 e ρ_2. Podemos observar que cada nó das duas páginas possui um nó correspondente na árvore SOM. Além disso, é possível observar que cada nó da árvore SOM possui um valor de frequência, indicando a quantidade de páginas que possuem aquele nó em específico. Além disso, cada nó está ligado a uma lista contendo as páginas que possuem conteúdo textual naquele nó.

Para ilustrar, o processo de criação de uma árvore SOM será descrito a partir das páginas ρ_1 e ρ_2 da Figura 3.3. Começando pela raiz dessas páginas, é possível verificar que ambas possuem a tag $HTML$, e portanto esta mesma tag será adicionada na árvore SOM com frequência igual a 2 e lista de páginas vazia, uma vez que ambas as páginas não possuem conteúdo textual nesta tag. Seguindo um caminhamento pré-ordem, verificamos que ambas as páginas também possuem o nó *BODY*, que também será adicionando na árvore SOM com frequência igual a 2 e lista de páginas vazia. Já as tags *DIV* à esquerda de ambas as páginas possuem conteúdo textual e, portanto, o nó *DIV* à esquerda da árvore SOM deverá possuir uma lista contendo as páginas, assim como frequência igual a 2. É possível observar que as tags da árvore SOM possuem os mesmos *labels* das tags correspondentes nas árvores DOM.

Outros exemplos são as folhas *PRE* e *P*, que ocorrem apenas nas páginas ρ1 e ρ2 respectivamente. Na árvore SOM essas duas tags são representadas com frequência igual a 1, e lista de páginas contendo apenas a página em que cada uma ocorre.

Fernandes et al [13] demonstraram que evidências específicas da árvore SOM possuem informações importantes para o processo de segmentação de páginas. Por exemplo, os autores identificaram que os nós que ocorrem em poucas páginas de um mesmo site

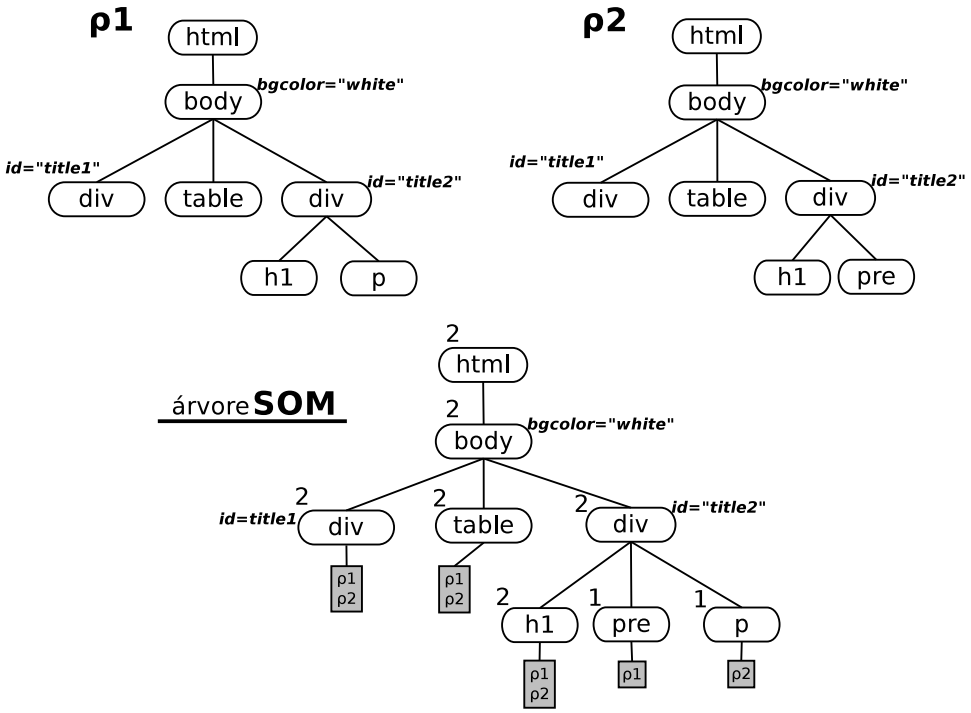

Figura 3.3: Exemplo de construção da árvore SOM

tendem a ser fragmentos internos de blocos, e que os nós com maior frequência tendem a fazer parte do *label* dos blocos. Exemplos dessas tags de menor frequência são encontradas em tabelas, links e figuras que podem ser usadas para complementar o conteúdo de um bloco em algumas páginas. Assim, assume-se que tags que ocorrem de forma irregular entre as páginas são partes internas dos blocos, e não são capazes de definir as estruturas das páginas.

3.2 Passo 3: Identificação dos Blocos através da Árvore SOM

Nesta seção descreveremos o terceiro passo do processo de segmentação conforme proposto por [13]. A idéia básica dessa abordagem é identificar e remover nós da SOM que, segundo a avaliação de especialistas humanos, sejam pertencentes ao interior de um ou mais blocos. A idéia é deixar na SOM apenas nós dispostos entre as raizes das páginas e a raízes dos blocos (isto é, nós que pertencem ao label de um ou mais blocos - vide seção 2.1). Dessa forma, cada nó folha da árvore SOM será uma referência para a raiz de um conjunto de blocos com o mesmo label e pertencentes a páginas diferentes.

Neste passo a árvore SOM é percorrida em duas fases. O primeiro apresentado na seção 3.2.1, a árvore é percorrida em um percurso pré-ordem onde verifica-se um possível aninhamento de conteúdo. No segundo passo, apresentado na seção 3.2.2 a árvore SOM é percorrida em pós-ordem e a frequência de cada nó é verificada com base em um limiar.

3.2.1 Poda de nós com conteúdo aninhado

Ao inserir diversas páginas em uma árvore SOM, é possível que alguns nós possuam conteúdo textual em algumas páginas e ao mesmo tempo sejam nós internos em outras páginas, como no exemplo da Figura 3.4 que apresenta um trecho de árvore SOM com conteúdo aninhado após a inserção de uma terceira árvore DOM. Esta situação de conteúdo aninhado na árvore SOM pode ser causada por dois motivos.

O primeiro motivo se deve a pequenas diferenças nas árvores DOM de páginas com estruturas semelhantes levando conteúdos que deveriam estar em um mesmo nó folha a estarem em folhas diferentes. O segundo motivo para essa condição de aninhamento da árvore SOM ocorre quando os nós que compõem o rótulo de um bloco de uma página estão contido no rótulo de um bloco de outra página com estrutura diferente.

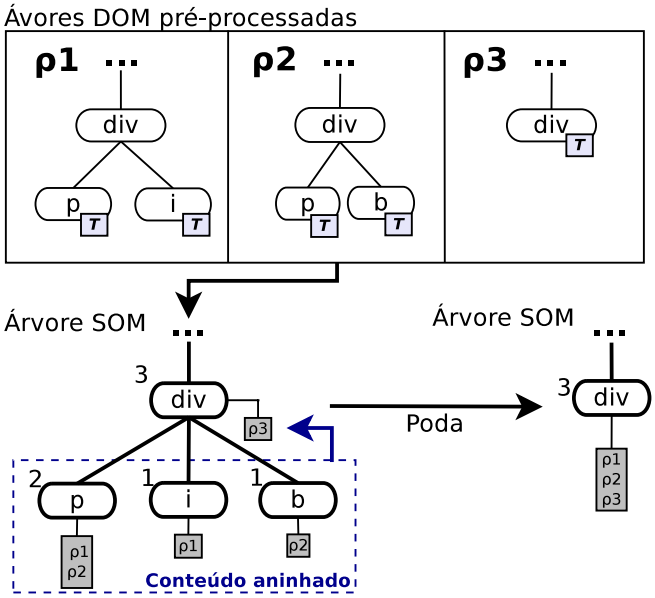

Figura 3.4: Conteúdo aninhado após a inserção da página ρ3 na árvore SOM

Essa primeira etapa de poda na árvore SOM visa eliminar aninhamento de conteúdo decorrentes do primeiro motivo citado. Enretanto, devido a segunda possível razão para aninhamentos, não podemos assumir que todo conteúdo aninhado deve ser removido da

árvore SOM. Em uma tentativa de evitar que a poda seja realizada em casos decorrentes da segunda razão, foi definida uma distância máxima de aninhamento para ocorrer a poda. A distância é medida pela profundidade entre os nós com conteúdo, e α define a distância máxima para que os nós sejam considerados próximos. Em [13] os autores adotaram o valor 6 como distância máxima para poda.

3.2.2 Poda de nós pela frequência

Neste segundo passo do processo de poda da árvore SOM, foi estabelecida uma frequência mínima desejada para os nós, sendo assim, ao atingir um nó *N* o processo de poda verifica se todos os seus filhos possuem frequência abaixo de um dado limiar β. Quando todas as frequências são inferiores ao limiar, as listas de documentos de cada nó abaixo de *N* são combinadas a lista de documentos do nó *N*, em seguida, todos os nós descendentes de *N* são podados da árvore. A regra de podar apenas quando todos os irmãos possuírem frequência abaixo do limiar é tão somente para evitar que sejam gerados conteúdos aninhados, como consequência, permite que em alguns casos elementos com frequência menor que o limiar permaneçam na árvore.

Na figura 3.5 há um exemplo deste processo de poda utilizando o limiar limiar $\beta = 8$ proposto em [13]. Observe que no trecho de árvore SOM do quadro 1 da figura há um conjunto de nós irmãos com frequência abaixo do limiar estabelecido. Esses elementos são portanto removidos da árvore. No quadro 2 da mesma figura é ilustrado o mesmo trecho de árvore SOM após o processo de poda descrito.

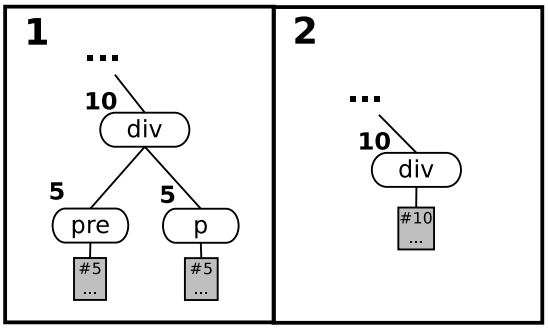

Figura 3.5: Exemplo de poda sobre a árvore SOM com $\beta = 8$

Diferente dessa estratégia de poda da árvore SOM utilizada em [13], propomos um processo mais robusto para este ultimo passo do método de segmentação automática. No capítulo 3.3 apresentamos esta nova estratégia de poda da árvore SOM, que utiliza

técnicas de aprendizagem de máquina para definir regras e limiares.

3.3 Aprendendo a Segmentar

O terceiro passo do método adotado como *baseline* consiste em identificar e remover da árvore SOM elementos que fazem parte da estrutura interna dos blocos, mantendo apenas o conteúdo textual associado a eles. Em nosso trabalho utilizamos técnicas de aprendizagem de máquina para alcançar melhores resultados neste processo de poda. Sendo assim, neste Capítulo apresentamos esse processo a partir da utilização de um classificador binário, que deve decidir quais nós da árvore SOM são internos a blocos e portanto devem ser podados da árvore SOM. A classificação avalia conjuntos de elementos da árvore SOM, na Seção 3.3.1 apresentamos a regra deste passo que permite identificar se um conjunto de nós da árvore SOM é candidato à classificação, devendo portanto ser representado como uma instância para o método de aprendizagem de máquina. Na Seção 3.3.2, demonstramos a estratégia de poda da árvore SOM. Por fim, na Seção 3.3.3, apresentamos o processo de extração de amostras que resulta em um conjunto de instâncias para a base de treino que é dada como entrada ao método de AM.

3.3.1 Evidências

Para gerar os classificadores para o processo de segmentação automática, utilizamos técnicas de aprendizagem supervisionada onde um conjunto de exemplos (instâncias rotuladas) é utilizado para gerar uma função capaz de classificar novas instâncias ainda não rotuladas. Sendo uma instância um ponto em um espaço N-dimensional onde N corresponde à quantidade de evidências usadas para representar a instância, o objetivo da função gerada no processo de AM é classificar novos pontos neste espaço.

Neste trabalho, uma instância representa um conjunto de nós da árvore SOM que são candidatos a classificação sendo esse conjunto chamdo de amostra válida. Um dado conjunto de nós é considerado uma amostra válida apenas quando: (i) É formado por um nó interno e todos os seus filhos e (ii) todos os filhos são folha da árvore SOM. Por exemplo, na Figura 3.6, o elemento *div* à direita e seus filhos $h1$, *pre* e *p* formam uma amostra válida. A identificação das classes de blocos dá-se através de sucessivas classificações das amostras entre as categorias B de elementos que pertencem ao mesmo

bloco e devem ser podados ou *NB* elementos que não pertencem ao mesmo bloco e devem ser mantidos na árvore.

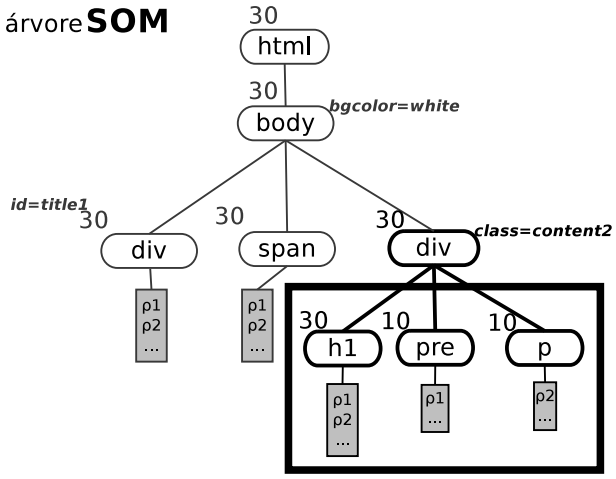

Figura 3.6: Exemplo de amostra válida a ser classificada

Para o processo de classificação, cada amostra válida é representada por um conjunto de evidências previamente definidas que descrevem as características da amostra. Esse conjunto de evidências constitui uma instância e é com base nos valores dessas evidências é que o classificador associa cada instância a uma das classes, por esse motivo é interessante utilizarmos evidências que tenham relação com o problema a ser tratado. Neste sentido, nos baseamos em métodos de segmentação encontrados na literatura para definir parte das evidências que foram utilizadas neste trabalho.

A seguir são apresentadas as 23 evidências definidas para representar cada conjunto de elementos a ser classificado nesta etapa da segmentação.

1. **NumeroDeElementos**: essa evidência possui como valor a quantidade de elementos (nós) que formam a instância.

2. **DistanciaRaiz**: essa evidência corresponde à quantidade de arestas entre os elementos da instância e a raiz da árvore SOM.

3. **MaisComum**: a frequência de um nó na árvore SOM é no máximo igual à frequência do nó pai. Essa evidência corresponde à razão entre a frequência do nó pai e a maior frequência entre os nós filhos.

4. **MenosComum**: essa evidência corresponde à razão entre a frequência do nó pai e a menor frequência entre os nós filhos.

5. **PaginasComConte**: razão entre o total de páginas que os nós filhos ocorrem com conteúdo textual e a frequência do nó pai.

6. **AncComConte**: essa evidência corresponde à quantidade de arestas entre os elementos que formam a instância e um ancestral com conteúdo textual.

7. **PopularidadePai**: corresponde à razão entre a frenquência do nó pai da amostra e a quantidade de páginas utilizadas para gerar a árvore SOM.

8. **QuantPaiCont**: essa evidência é a razão entre a quantidade de páginas em que o nó pai ocorre com conteúdo textual e o total de páginas em que ele ocorre na árvore SOM (frequência).

9. **QuantidadeTextual**: essa evidência é a razão entre a quantidade de conteúdo textual de todos os elementos que formam a instância e o número de nós na instância.

10. **TodosComConte**: é uma evidência booleana que assume o valor *verdade* quando todos os nós da instância possuem conteúdo textual.

11. **PaiBlock**: as tags podem ser divididas em dois tipos: *inline* e *block*. Tags *inline* são tags que mantém o fluxo comum do conteúdo e tags *block* quebram o fluxo e geram o elemento abaixo do último *block*. Essa é uma evidência booleana que assume o valor *verdade* quando o nó pai da amostra é uma tag do tipo *block*.

12. **PossuiTagsBlock**: essa evidência booleana informa se há entre os nós filhos dos elementos que formam a instância alguma tag do tipo *block* como: *div*, *table* e *h*1.

13. **PossuiTagsInline**: essa evidência booleana informa se há entre os nós filhos dos elementos que formam a instância alguma tag do tipo *inline* como: *td*, *img* e *textarea*..

14. **TagsIguais**: essa evidência booleana assume valor *verdade* quando a instância é formada por um conjunto de nós generalizados. Nós que possuem o mesmo pai, possuem o mesmo tipo de tag e são adjacentes.

15. **PaiTiosIguais**: essa evidência assume o valor *verdade* quando o nó pai entre os elementos que formam a instância pertence a um conjunto de nós generalizados.

16. **TioComFilhos**: Essa evidência assume o valor *verdade* quando o nó pai dos elementos da instância possui pelo menos um nó irmão não folha.

17. **TioFolha**: essa evidência assume o valor *verdade* quando o nó pai dos elementos da instância possui pelo menos um nó irmão folha.

18. **TioBlock**: essa evidência booleana informa se há entre os nós adjacentes ao nó pai da amostra pelo menos um nó do tipo *block*.

19. **TioInline**: essa evidência booleana informa se há entre os nós adjacentes ao nó pai pelo menos um nó do tipo *inline*.

20. **TioComConteudo**: essa evidência assume o valor *verdade* quando houver algum elemento com conteúdo textual entre os nós adjacentes ao elemento pai da amostra.

21. **QuantTios**: o valor dessa evidência corresponde à quantidade de nós adjacentes ao nó pai dos elementos que formam a instância.

22. **QpccIgualpp**: essa evidência verifica se a quantidade de páginas com conteúdo textual na amostra é igual à popularidade do pai. É *verdade* quando todos os nós da instância ocorrem com conteúdo textual em todas as páginas nas quais o nó pai da instância ocorre.

23. **TagsTipBlock**: o valor dessa evidência é a razão entre o número de tags *block* e o total de tags na instância.

Além das evidências citadas, há um outro atributo que compõe a instância. Chamado de *MesmoBloco*, esse atributo corresponde ao valor alvo ou à classe da instância. Pode variar entre: *B*, nós que devem ser podados (são elementos internos a blocos) ou *NB*, nós que não devem ser podados (cada nó corresponde a uma classe de blocos).

3.3.2 Estratégia de Segmentação

Para descrever nossa estratégia de segmentação, vamos utilizar como exemplo a Figura 3.7, que mostra passo a passo como ocorre o processo de identificação das classes de blocos. Primeiro, devem ser identificados os elementos folhas da árvore SOM. Na Figura 3.7, esses elementos são destacados com uma borda mais escura. Esses elementos serão lidos da esquerda para a direita pelo algoritmo em questão.

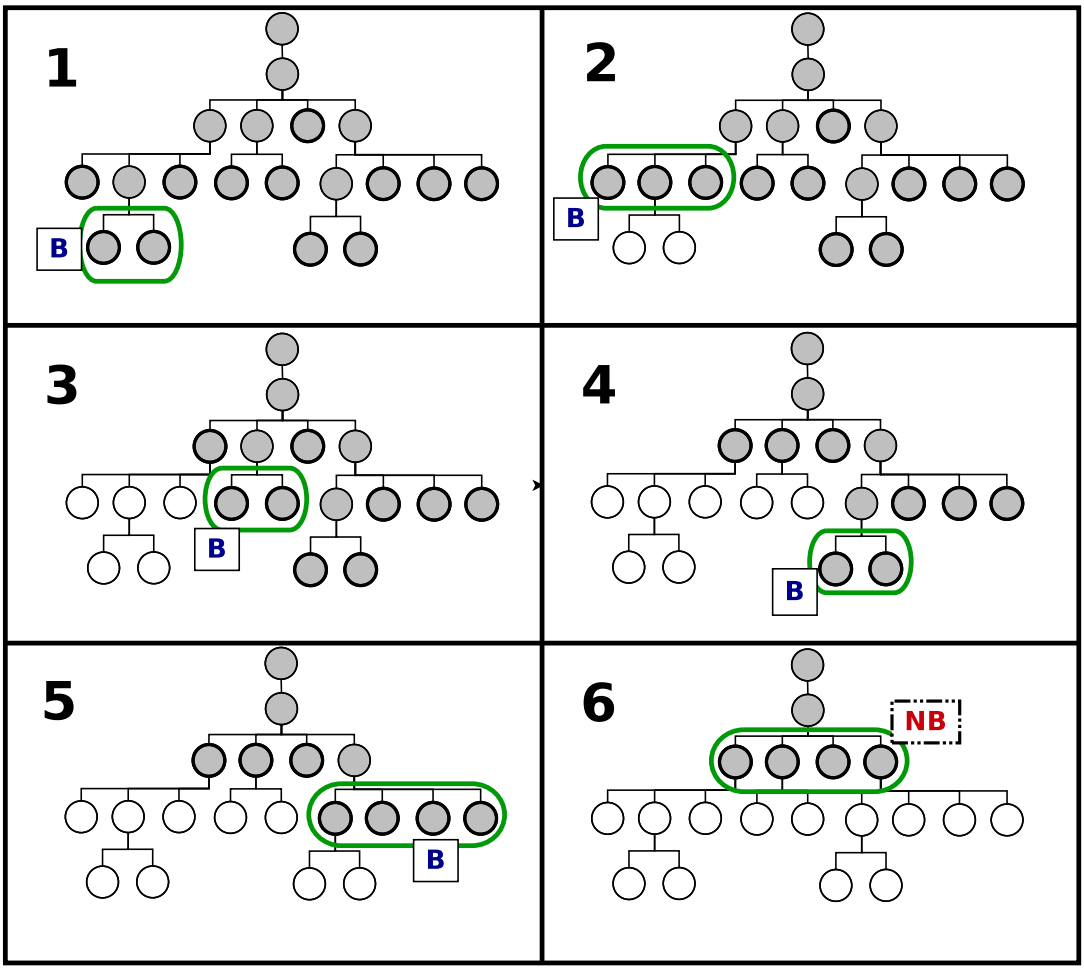

Figura 3.7: Passo a passo do processo de segmentação

Na árvore do quadro 1 dessa Figura, o primeiro elemento folha (o nó mais à esquerda da árvore) possui um irmão não folha, não sendo portanto um candidato à amostra. No entanto, os irmãos do segundo elemento são todos folhas da árvore SOM, de forma que tais elementos formam uma amostra válida para nossa estratégia de segmentação. Conforme pode ser verificado no quadro 1, essa primeira amostra pertence à classe *B* (seus elementos pertencem a uma mesma classe de blocos). Por esse motivo, os elementos dessa primeira amostra passam a ser representados pelo nó pai no quadro 2 da Figura 3.7, sendo este pai o próximo elemento a ser considerado por nosso algoritmo. No quadro 2, observe que o novo nó com conteúdo também formou uma amostra da classe *B*. Esse processo se repete até que todos os elementos com conteúdo textual sejam verificados.

O último passo da segmentação na Figura 3.7 ocorre no quadro 6, onde a amostra encontrada é classificada como *NB*. Neste momento, cada elemento folha com conteúdo textual que forma a amostra é entendido como uma classe de blocos do Web site. Por

não haver mais elementos a serem verificados, o processo de segmentação se conclui, resultando em uma árvore SOM com três classes de blocos.

3.3.3 Extração de Instâncias para Treino

A construção da coleção de instâncias rotuladas para a base de treino consiste em utilizar informações preveamente adicionadas por especialistas em um processo manual de identificação dos blocos de cada página. A construção da base de treino ocorre em um processo bastante similar ao de segmentação apresentado na seção anterior, se diferenciando em dois pontos importantes que são: (i) A utilização de um classificador denominado oráculo que utiliza informções da segmentação manual para identificar se a amostra é formada por elementos que pertencem a um mesmo bloco ou elementos que pertencem a blocos distintos. E (ii) durante o processo, cada instância classificada pelo oráculo é adicionada a um arquivo que constitui a base de treino.

Vamos utilizar o Algoritmo 1 para descrever essa fase de identificação das instâncias e seus respectivos rotulos para a construção da base de treino. O primeiro passo consiste em identificar os elementos folhas da árvore SOM conforme é apresentado na linha 1 do Algoritmo 1.

Algoritmo 1 Pseudo código do processo de poda da árvore SOM.

```
 1: Lista < Elementos > lista ← getFolhas(SOM);
 2: for (j = 0; j < lista.size ; j++) do
 3:     estrategia(lista[j])
 4: end for
 5: function ESTRATEGIA(Elemento n)
 6:     if (saoFolhas(n.Irmaos)) then
 7:         Amostra a[ ] ← ExtrairCaracteristicas(n.Irmaos);
 8:         categoria ← mesmoBloco(n.irmaos);
 9:         base.add(a,categoria);
10:         if (categoria == "B") then
11:             removeDaLista(n.irmaos)
12:             n.pai ← getConteudo(n.irmaos)
13:             estrategia(n.pai)
14:         end if
15:     end if
16: end function
```

No passo seguinte, uma lista dos elementos encontrados será passada como parâmetro para a função *estratégia*, que é responsável por identificar as amostras e suas categorias.

Dentro dessa função, o primeiro passo é identificar se o nó faz parte de uma amostra válida para o nosso experimento, isso é, se os irmãos deste nó são folha (linha 6, Algoritmo 1). Quando o nó não faz parte de uma amostra válida, a função termina e o próximo elemento da lista é chamado. Caso contrário, as características da amostra (composta pelo nó e seus irmãos) são extraídas (linha 7, Algoritmo 1). Em seguida, a categoria da amostra é verificada (linha 8, Algoritmo 1) e então, o vetor de características e a categoria correspondente são adicionadas à base.

A identificação da categoria de uma amostra consiste em verificar se todos os elementos da amostra são internos à mesma classe de blocos na árvore. Os próximos elementos a serem verificados dependem da categoria da última amostra. Quando a amostra é da categoria *B*, o nó pai dos elementos da amostra passa a representá-los e será o próximo nó a ser verificado (Algoritmo 1, linhas 10-14). Caso a última amostra seja da categoria *NB*, a função termina e o próximo elemento do vetor lista é chamado. O processo continua enquanto houver elementos na lista.

Voltando à análise das amostras associadas a categoria *B*, observe na linha 11 (Algoritmo 1) que a *lista* é manipulada. Este passo remove da *lista* possíveis ocorrências de elementos que pertencem à amostra verificada e fora podados da árvore.

Capítulo 4

Experimentos

Com foco em implementar uma versão mais robusta do *baseline*, neste trabalho substituímos as funções criadas de maneira *ad hoc* utilizada no último passo do *baseline* por funções geradas através de técnicas de mineração de dados e aprendizagem de máquina. Sendo assim, o objetivo da realização dos experimentos foi comparar os resultados reportados no *baseline* com os obtidos com a implementação do nosso método. Para isso, foram utilizadas as mesmas métricas de avaliação (Seção 2.3) e coleções de páginas Web do *baseline*.

Neste capítulo apresentamos as coleções sobre as quais os experimentos foram realizados e em seguida apresentamos e discutimos os resultados obtidos a partir da comparação entre o *baseline* e o método proposto.

4.1 Coleções de Páginas Web

Para a aplicarmos as técnicas de aprendizagem de máquina no contexto de segmentação de páginas Web deste trabalho, é fundamental que haja coleções de páginas segmentadas manualmente para serem utilizadas na etapa de treino e teste do processo de aprendizado supervisionado. Essas coleções também serão usadas para avaliar a qualidade da segmentação obtida através do método proposto, bem como através do *baseline*.

Para os experimentos deste trabalho, foram utilizadas as mesmas coleções utilizadas para realização dos experimentos em [13]. Temos então quatro diferentes coleções: CNN, IG, CNET e BLOGS. A coleção CNN contém 16.257 páginas com conteúdo em inglês

Tabela 4.1: Distribuição das páginas por coleção.

Coleção	Site	Páginas	
IG	News	26,466	www.ultimosegundo.com.br
	Forum	6,389	www.jornaldedebates.com.br
	Recipe	1,605	www.panelinha.com.br
	Total	**34,460**	
CNN	News site	16,257	www.cnn.com
	Total	**16,257**	
CNET	News	131,474	www.news.com
	Downloads	99,186	www.downloads.com
	Reviews	64,142	reviews.cnet.com
	Shopper	57,968	www.shopper.com
	Total	**352,770**	
BLOGs	Boing Boing	14,173	www.boingboing.net
	CNET	8,054	news.cnet.comtech-blogs
	Engadget	6,343	www.engadget.com
	Gizmodo	4,454	us.gizmodo.com
	Google	1,050	googleblog.blogspot.com
	Life Hacker	3,997	www.lifehacker.com
	Mashable	7,410	www.mashable.com
	Slash Film	5,376	www.slashfilm.com
	Tech Crunch	3,198	www.techcrunch.com
	Total	**54,055**	

coletadas do site de notícias da rede CNN. A coleção IG, coletada de três sites[1], contém 34.460 páginas em português. A coleção CNET foi coletada de quatro sites[2] e contém 352.770 páginas em inglês. Por fim, a coleção BLOGS possui 54.055 páginas em inglês coletadas dos nove blogs mais populares[3]. Na Tabela 4.1 essas coleções estão dispostas de forma mais detalhada.

Todas estas coleções foram segmentadas de maneira semi-supervisionada com o auxílio de um especialista, conforme descrito em [13]. Primeiro as páginas foram manualmente agrupadas de acordo com a similaridade de sua estrutura interna. Em seguida, foi utilizado o algoritmo VIPS para segmentar as páginas de cada agrupamento. Os parâmetros do VIPS foram selecionados manualmente para cada agrupamento de páginas. Por fim, a segmentação realizada pelo VIPS foi verificada por um usuário especialista que, quando não concordava com o resultado, adaptava a divisão de blocos de acordo com a sua percepção. Como houve uma forte intervenção manual neste processo semi-automático, dizemos que essa segmentação é manual.

[1] sites coletados do portal www.ig.com.br

[2] sites coletados do portal wwww.cnet.com

[3] segundo um ranking disponível em *technorati.com*

4.2 Gerando Bases Para Treino e Teste

O método automático é avaliado de acordo com sua capacidade de alcançar resultados similares aos obtidos através da segmentação manual. Sabendo disso, para gerar as bases de treino e de teste, as coleções de páginas segmentadas manualmente são utilizadas como referência. Ao combinar as páginas de um site em uma árvore SOM cada nó recebe um contador (*blocCont*) cujo valor corresponde à quantidade de páginas em que o nó foi considerado raiz de bloco na classificação manual das páginas HTML. O valor desse atributo contador é usado por um classificador exclusivo para extração de informações para as bases de treino e teste.

Basicamente o processo de criação das bases de treino e teste utilizam um classificador que verifica se os elementos que formam a amostra são descendentes de um nó com *blocCont* maior que zero. Quando isso for verdade, a amostra é associada à categoria *B* de elementos que devem ser combinados, caso contrário é associada à categoria *NB*. Após a classificação cada amostra é adicionada como um exemplo à base de treino. Caso haja mais de um nó com *blocCont* no caminho até a raiz da árvore SOM, apenas o nó com maior valor de *blocCont* é considerado.

A base de treino com instâncias corretamente rotuladas foi usada como entrada ao SVM, que ao final do processo de aprendizagem gera um função ou modelo do conhecimento que é usado para classificar novas instâncias. Para avaliação do modelo do conhecimento é preciso uma outra base com exemplos extraídos de outros Web sites. Os rótulos atribuídos pelo oráculo na base de teste são usados como gabarito para avaliar a qualidade da classificação com do modelo em questão.

Neste trabalho, criamos quatro bases de teste, sendo uma para cada coleção como apresentado na Tabela 4.2. Também foram criadas três bases para treino combinando as coleções em duplas [CNN,IG] [CNN,BLOGS] e [IG,BLOGS]. Nesta mesma tabela é apresentada a quantidade de intâncias por classe em cada uma das bases. Observe que há um desbalanceamento entre as classes de todas as bases. Para evitar que esse desbalanceamento pudesse influenciar na construção do modelo do conhecimento, aplicamos pesos às amostras da classe com menos elementos (*NB*). A quantidade de elementos após a aplicação dos pesos é apresentada na coluna mais à direita da tabela.

Tabela 4.2: Quantidade de instâncias por classe nas bases de treino e teste utilizadas

	Classe B	Classe NB	pesoNB	Classe NB
IG	746	275	*1	275
CNN	1.894	548	*1	548
BLOGS	1.824	162	*1	162
CNET	10.065	1.683	*1	1.683
CNN+IG	2.640	823	*3	2.469
CNN+BLOGS	3.718	710	*5	3.550
IG+BLOGS	2.570	437	*5	2.185

4.2.1 Ajuste dos parâmetros de treino

Como apresentado na seção 2.2.2, a técnica de SVM depende da escolha da função de kernel e seus respectivos parâmetros. Entre as funções de kernel testadas para nosso problema, a que apresentou melhor resultado foi a polinomial. Esta função é apresentada na Equação 4.1.

$$k(x_i, x_j) = (\gamma * x_i * + x_j)^d \tag{4.1}$$

A função polinomial depende do ajuste dos parâmetros gamma(γ), grau(d) e coef0(R). O processo de ajuste desses parâmetros se dá através da utilização de duas bases de amostras rotuladas, sendo uma para treino e a segunda para testes do classificador. Então, são realizados vários processos de criação e avaliação de classificadores enquanto os valores desses parâmetros são ajustados para identificar o ajuste com melhor desempenho.

Para demonstrar o processo de ajuste dos parâmetros do kernel polinomial, apresentamos na Tabela 4.3 o desempenho de classificadores gerados de uma mesma base de treino [CNN,BLOGS] e avaliados sobre as amostras da base de teste [IG]. Cada linha da tabela corresponde à avaliação de um classificador que se diferencia dos demais pelo valor de parâmetro apresentado na primeira coluna.

Os resultados da tabela estão ordenados por parâmetro. Primeiro apresentamos o resultado dos classificadores quando variamos o parâmetro d, em seguida são apresentados os resultados quando variamos o parâmetro γ e por fim, apresentamos os resultados ao variar o parâmetro R. Nos exemplos apresentados na tabela, já estão sendo considerados os valores que foram utilizados em nosso trabalho. Por exemplo, ao variar o parâmetro d os parâmetros R e γ já estão ajustados.

As métricas usadas para avaliação dos classificadores são descritas com mais detalhes

Tabela 4.3: Identificação do melhor valor para o parâmetro Grau do Kernel polinomial

Grau (d)	Acurácia	Revocação B	Revocação NB	Tempo(s)
1	0,70	0,73	0,63	982,86
2	0,30	0,05	0,99	982,86
3	0,70	0,72	0,66	8,43
4	0,56	0,63	0,36	57389,52
5	0,27	0,00	1,00	10,67
Gamma (γ)	Acurácia	Revocação B	Revocação NB	Tempo(s)
100	0,32	0,08	0,96	939,16
10	0,31	0,01	0,96	943,11
1	0,37	0,16	0,94	1035,70
-1	0,70	0,72	0,66	8,45
-10	0,70	0,72	0,66	8,61
-100	0,70	0,72	0,66	8,43
Coef0 (R)	Acurácia	Revocação B	Revocação NB	Tempo(s)
0	0,35	0,36	0,34	11,46
1	0,35	0,36	0,34	10,30
2	0,70	0,72	0,66	8,10
3	0,70	0,72	0,66	8,95
4	0,70	0,72	0,66	9,06
5	0,70	0,72	0,66	8,96
10	0,70	0,72	0,66	8,12

na proxma seção. Basicamente os valores de revocação e acurácia, demonstram o quanto o classificador acertou no processo de classificação das amostras da base de teste. Sendo a acurácia o desempenho para toda a base de teste e a revocação o desempenho para cada categoria. A coluna mais à direita na tabela possui o tempo total em segundos do processo de aprendizado e de elaboração de cada classificador.

Ao final do processo de ajuste dos parâmetros, consideramos utilizar os seguintes valores de configuração do kernel polinomial: γ com valor -100, d com valor 3 e R com valor 10. Esses mesmos valores foram identificados como melhor ajuste também para as outras bases de treino.

4.2.2 Criação e teste dos classificadores gerados com SVM

Nesta subseção, apresentamos os resultados de teste dos modelos do conhecimento obtidos com o SVM, bem como os resultados obtidos com uma função baseada nos parâmetros α e β utilizados no *baseline*.

Após fornecermos as bases de treino (Seção 4.2) como entrada para a ferramenta SVMTrain da biblioteca LibSVM, obtivemos três classificadores. Cada base de treino é formada por instâncias extraídas de duas coleções, portanto, para testar cada classificador

utilizamos uma base de instâncias extraídas de uma terceira coleção. Para cada base de teste avaliamos também o desempenho de uma função baseada nos limiares do *baseline* para podermos comparar os resultados.

Para medir a qualidade dos classificadores calculamos a *Acurácia*, *Revocação em B* e *Revocação em NB* a partir dos resultados obtidos com um classificador em uma base de teste. A acurácia é responsável por medir o quanto o classificador se aproximou dos resultados de parâmetro da base de teste. O valor é obtido a partir da razão entre a quantidade de instâncias classificadas corretamente e o total de instâncias existentes na base. Vamos utilizar os resultados obtidos com a base de teste IG (Tabela 4.4) para exemplificar o cálculo dessa medida. Na Tabela 4.2 é possível verificar que a base de teste IG possui 1121 instâncias, sendo 746 da categoria *B* e 275 da categoria *NB*. Na Tabela 4.4, a acurácia alcançada com o classificador $CNN + BLOGS$ (CB) nessa mesma base foi de $0,70$. Multiplicando esses dois valores obtemos a quantidade de instâncias corretamente classificadas por *CB* na base de teste *IG*. O resultado então é de $0,70 * 1121 = 785$ amostras corretamente classificadas.

A revocação em *B* ou *NB* é calculada de forma semelhante à acurácia, porém são consideradas apenas as instâncias das respectivas classes. A revocação de *B* é obtida da razão entre as instâncias classificadas como *B* e o total de instâncias da categoria *B* na base de teste.

Tabela 4.4: Teste dos classificadores gerados a partir das bases de treino e o classificador utilizado no *baseline* com limiares $\alpha = 6$ e $\beta = 8$.

Classificador	Teste	Acurácia	Revocação B	Revocação NB
CNN+Blogs	IG	0,70	0,72	0,66
FBaseline	IG	0,41	0,50	0,17
IG+Bloggs	CNN	0,59	0,62	0,51
FBaseline	CNN	0,48	0,55	0,22
CNN+IG	BLOGS	0,47	0,45	0,72
FBaseline	BLOGS	0,65	0,69	0,25
CNN+IG	CNET	0,47	0,48	0,44
FBaseline	CNET	0,78	0,89	0,10

Analisando os resultados reportados na Tabela 4.4 podemos observar que os classificadores obtidos com aprendizagem de máquina alcançaram maior acurácia em duas das quatro bases de teste. No entanto, para todas as bases de teste o maior valor de revocação na categoria *NB* foi obtido com os métodos de aprendizagem de máquina.

Um ponto importante em AM refere-se ao desbalanceamento de classes. Por exemplo,

a base de teste da coleção *BLOGS*, possui 1986 instâncias sendo 1824 da categoria *B* (Tabela 4.2). Um classificador simples que classifique sempre novos exemplos como pertencentes à classe majoritária *B* teria uma acurácia de $0,9184$. Apesar da boa acurácia, esse resultado seria indesejável pois é crucial para o problema de poda da árvore SOM que o classificador saiba quando parar de combinar os elementos.

Os resultados de teste apresentados até então foram obtidos a partir de treinos com todas as evidências descritas na seção 3.3.1. Para demonstramos o impacto de cada uma das evidências, realizamos o processo de treino e teste com cada uma. Ou seja, para cada dupla de coleções definidas para treino foram gerados 23 classificadores, cada um considerando apenas uma das evidências. Para avaliar esses novos classificadores, utilizando a mesma relação entre treino e teste apresentada na Tabela 4.4. Por exemplo, cada classificador gerado a partir da dupla [CNN,IG] foi testado sobre a base de exemplos extraídos da coleção BLOGS.

Tabela 4.5: Desempenho dos classificadores por cada evidência.

		CNN+IG→Blogs			**IG+Blogs→CNN**		
CLASSIFICADOR	ID	Acurácia	Revoc. B	Revoc. NB	Acurácia	Revoc. B	Revoc. NB
nElementos	1	0,33	0,30	0,70	0,34	0,25	0,65
distanciaRaiz	2	0,70	0,69	0,84	0,48	0,40	0,76
***maisComum**	3	0,08	0,00	1,00	0,22	0,00	1,00
***menosComum**	4	0,08	0,00	1,00	0,22	0,00	1,00
***paginasComConte**	5	0,92	1,00	0,00	0,22	0,00	1,00
ancComConte	6	0,65	0,65	0,67	0,58	0,61	0,46
popularidadePai	7	0,48	0,49	0,38	0,41	0,41	0,39
quantPaiCont	8	0,25	0,19	0,90	0,71	0,86	0,17
quantTextual	9	0,41	0,41	0,33	0,36	0,32	0,50
todosComConte	10	0,46	0,46	0,45	0,34	0,32	0,44
***paiBlock**	11	0,92	1,00	0,00	0,77	0,99	0,01
possuiTagsBlock	12	0,46	0,47	0,35	0,60	0,56	0,71
***possuiTagsInline**	13	0,91	0,99	0,00	0,77	0,99	0,01
tagsIguais	14	0,47	0,47	0,54	0,65	0,76	0,26
paiTiosIguais	15	0,67	0,70	0,35	0,61	0,77	0,09
TioComFilhos	16	0,43	0,44	0,40	0,63	0,71	0,33
TioFolha	17	0,35	0,31	0,71	0,47	0,49	0,41
***TioBlock**	18	0,92	1,00	0,00	0,78	1,00	0,01
TioInline	19	0,71	0,75	0,30	0,73	0,92	0,09
TioComConteudo	20	0,57	0,59	0,35	0,60	0,74	0,13
quantTios	21	0,28	0,24	0,82	0,66	0,78	0,25
qpccIgualpp	22	0,38	0,35	0,80	0,45	0,46	0,41
tagsTipBlock	23	0,46	0,47	0,35	0,60	0,57	0,71

Na Tabela 4.5 apresentamos os resultados dos classificadores gerados a partir das coleções [CNN,IG] e [IG,BLOGS] testados sobre as coleções BLOGS e CNN respectivamente. Cada linha da tabela corresponde a um classificador cujo processo de aprendizagem considerou apenas uma evidência, correspondente ao nome do classificador. Para cada coleção de treino há 23 classificadores avaliados a partir das medidas de acurácia e revocação em suas respectivas bases de testes.

Ao analisarmos os resultados dos testes identificamos um comportamento indesejável entre os seis classificadores destacados em negrito na tabela. Observamos nos resultados um comportamento similar entre esses classificadores em todas as bases de treino. Nos valores apresentados na tabela podemos observar que os resultados de teste com esses classificadores associam todas as amostras a uma única classe. Resultando em 100% de revocação para uma categoria e 0% para a outra.

Com base nos resultados obtidos, geramos novos treinos desconsiderando as evidências em questão. Ou seja, utilizando as 17 evidências restantes para o processo de aprendizagem. No entanto, como não foi obtido nenhum ganho sobre o treino com todas as evidências, optamos por manter os classificadores em que o processo de aprendizagem considerou todas as evidências.

4.3 Resultados da Segmentação

Após o processo de aprendizagem e teste utilizando a biblioteca LibSVM (Seção 2.2.3) o passo seguinte é a utilização dos classificadores para a segmentação de páginas Web. Nesta seção vamos apresentar os resultados da segmentação automática utilizando esses classificadores. Também apresentamos nesta seção o desempenho dos classificadores durante o processo de poda.

Como apresentado no Capítulo 3.3, o processo de identificação das classes de bloco na árvore SOM é feito a partir de sucessivas classificações entre combinar ou não combinar grupos de elementos (amostras). Também foi apresentado que os grupos de elementos avaliados são selecionados dinamicamente a partir da classificação da amostra anterior. Por esta razão demonstramos na Tabela 4.6 o comportamento dos classificadores a partir das quantidades de instâncias avaliadas durante o processo de segmentação de cada árvore SOM usada no experimento.

Como vimos na Figura 3.7 que mostra um passo a passo do processo de segmentação, sempre que um grupo de nós é combinado o próximo elemento a ser verificado é o nó pai do grupo. Se um grupo pertence à categoria *NB* é certo afirmar que qualquer amostra acima na árvore também pertencerá à categoria *NB*. Portanto, se esse mesmo grupo for erroneamente classificado como *B*, mais amostras da categoria *NB* terão que ser avaliadas. Observe então que são obtidas amostras a mais da categoria *NB* sempre que o classifi-

cador erra a classificação de uma amostra desta categoria. Por exemplo, observe que a quantidade de amostras *NB* avaliadas durante a segmentação com o método automático (Tabela 4.6) é superior a quantidade de amostras obtida pelo classificador de referência (oráculo) na Tabela 4.2.

Tabela 4.6: Amostras avaliadas pelos classificadores durante o processo de poda da árvore SOM.

Treino	Teste	B	NB	Total
CNN+Blogs	IG	578	362	940
FBaseline	IG	497	522	1.019
IG+Bloggs	CNN	1.112	944	2.056
FBaseline	CNN	1.223	1.482	2.705
CNN+IG	BLOGS	1.048	172	1.220
FBaseline	BLOGS	1.390	235	1.625
CNN+IG	CNET	6.165	2.729	8.894
FBaseline	CNET	9.136	3.633	12.769

Como apresentado na Seção 4.2.2, os classificadores gerados com técnicas de AM obtiveram resultado superior na medida de revocação para a categoria *NB*. Esse resultado se reflete na quantidade menor de amostras avaliadas pelos classificadores baseados em AM quando comparado à quantidade de amostras avaliadas pelo *baseline*, como pode ser observado na Tabela 4.6.

Ao analisar nossos resultados verificamos duas situações na relação entre a classificação de grupos de elementos e a segmentação da página: (i) nem sempre ao acertar a classificação de um grupo de nós o método acerta na segmentação da página e (ii) nem sempre ao errar a classificação de um grupo de nós o método erra na segmentação da página.

A primeira situação ocorre quando em um grupo de nós a ser avaliado, parte dos irmãos pertencem a um bloco diferente dos demais. Neste caso, em que os nós não devem ser combinados (classe *NB*) por não pertencerem todos ao mesmo bloco, mesmo que o classificador acerte ao classificá-lo como tal, o método estará errando na segmentação, pois estes elementos também não pertencem cada um a um bloco distinto.

Para demonstrar a segunda situação, vamos usar a Figura 4.1 como exemplo. É possível observar que no trecho de código do quadro 1 desta figura há um atributo *bloco="true"* em uma das tags $< td >$. Este atributo, adicionado por um especialista, marca um bloco formado pelo conteúdo textual $[Texto1, Texto2]$. Como há este atributo na tag $< td >$,

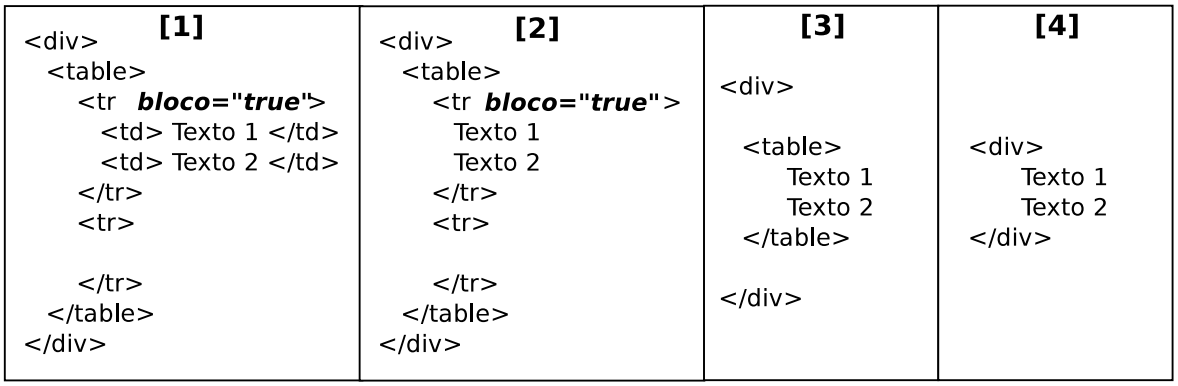

Figura 4.1: Exemplo que demonstra a possibilidade de sucesso na identificação de um bloco ainda que haja "erro"na decisão entre combinar ou não os elementos

sabe-se que ao verificar a tag $< table >$ com seus respectivos filhos $< td >$, o método deve classificar esta instância como *NB* deixando o código como no quadro 2 da figura. No entanto, em casos como deste exemplo, se o método classificador errar atribuindo classe *B* à instância (quadro 3 da figura), não haverá diferença na avaliação do segmentador para este bloco, pois o bloco continuará sendo formado pelos mesmos termos $[Texto1, Texto2]$. Da mesma forma, se o método errar mais uma vez e definir a tag $< div >$ como sendo o bloco (quadro 4 da figura), continuará não havendo erro na avaliação da segmentação deste bloco da página.

A segmentação automática pode ser aplicada à áreas em que uma segmentação próxima da percepção que o usuário tem como ideal seja necessária. Para avaliar essa proximidade utilizamos as métricas Índice Ajustado De Rand (AdjRand) e Informação Mútua Normalizada (NMI) apresentadas na Seção 2.3. Como referência, apresentamos também os resultados obtidos com o método proposto em [14].

Para cada método, com uma função gerada por aprendizagem de máquina e com uma função *adhoc* utilizando limiares fixos (*baseline*), foram calculados os valores de AdjRand e NMI para cada página das quatro coleções usadas nos experimentos. Na Tabela 4.7 são apresentadas as médias das duas métricas calculadas sobre as páginas de cada uma das coleções.

Os valores médios apresentados para as duas métricas mostram que o grau de concordância entre a segmentação automática com aprendizagem e a segmentação de referencia (manual) é elevado. Esse resultado comprova que é possível segmentar páginas de um site a partir de exemplos extraídos de outros sites.

Tabela 4.7: Valores médios de AdjRand e NMI sobre as páginas segmentadas com técnicas de AM e *baseline* nas 4 bases (IG, CNN, BLOGS e CNET).

		SVM	*Baseline*	Oráculo
IG	AdjRand	0,837	0,831	0,901
	NMI	0,886	0,895	0,945
CNN	AdjRand	0,918	0,849	0,943
	NMI	0,901	0,876	0,946
BLOGS	AdjRand	0,798	0,824	0,892
	NMI	0,863	0,866	0,925
CNET	AdjRand	0,681	0,500	0,831
	NMI	0,817	0,729	0,900

É interessante observarmos também a estabilidade dos classificadores. Observe que os valores médios de AdjRand e NMI obtidos com o *baseline* para as coleções CNN, IG e BLOG foram bastante próximos tendo uma variação de NMI entre $0,86$ e $0,89$ e os valores obtidos para CNET foram inferiores deixando a variação de NMI entre $0,72$ e $0,89$. Com as funções de aprendizagem obtemos maior estabilidade entre as médias de todas as coleções, tendo uma variação de NMI entre $0,81$ e $0,90$.

Nas Figuras 4.2 e 4.3 podemos observar o comportamento de cada método através das métricas AdjRand e NMI. Para cada métrica foram desenhados quatro gráficos, um para cada coleção (IG, CNN, BLOGS, CNET). Cada método tem seu desempenho em uma coleção evidenciado através de um curva no gráfico. Então, cada curva nos gráficos das duas figuras denota o desempenho de um dos métodos através de uma das métricas para uma determinada coleção.

Para traçar cada curva do método foram utilizados os cálculos da métrica para cada página da coleção segmentada, ordenando-as em ordem crescente pelo valor da métrica. Então para cada página da coleção marcamos uma coordenada (x, y) onde x representa a ordem das páginas e y o resultado da métrica calculada para a página.

Assim como observado na Tabela 4.7, os gráficos demonstram que a segmentação utilizando classificadores obtidos com técnicas de aprendizagem de máquina a partir de informações extraídas de outros sites, obteve um elevado grau de concordância com a segmentação manual. Através dos gráficos podemos entender melhor como cada coleção se comportou para cada método.

É possível verificar na Tabela 4.7 e nos gráficos das Figuras 4.2 e 4.3 que o novo método obteve ganho sobre o *baseline*, podemos observar ganho sobre as coleções CNN

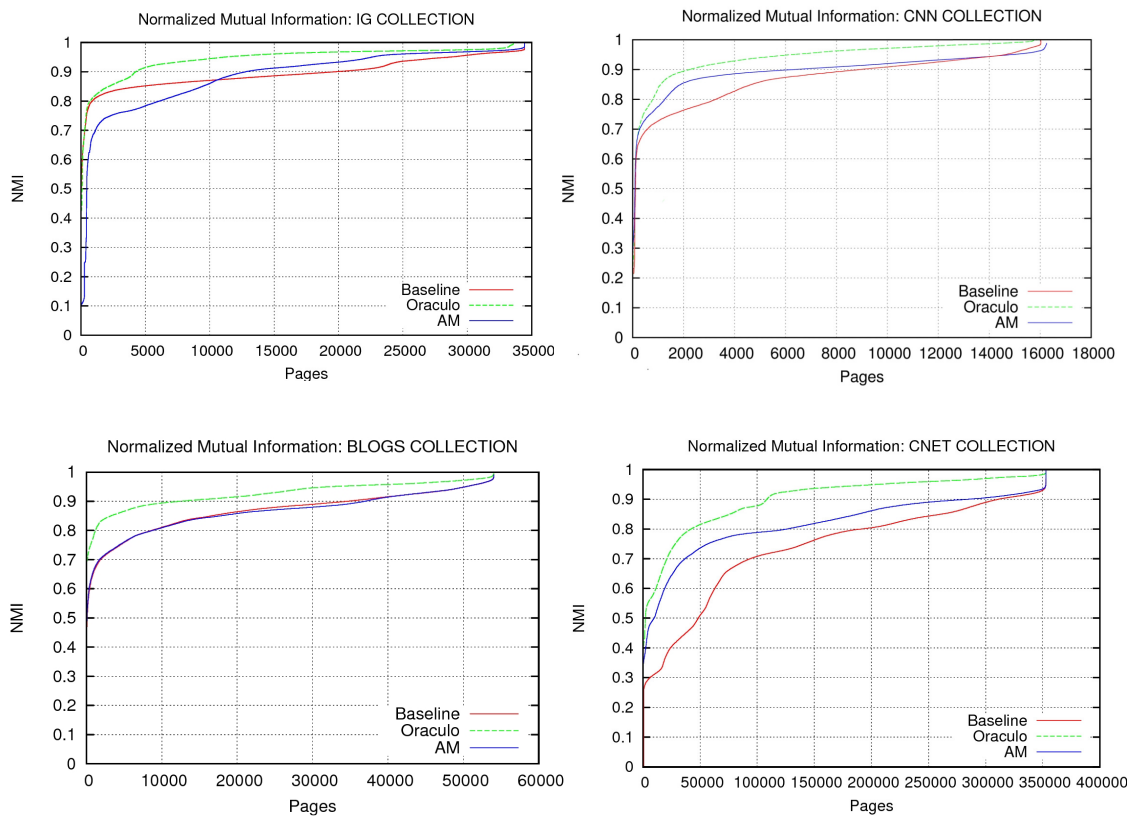

Figura 4.2: Avaliação das coleções IG, CNN, BLOGS e CNET com a métrica Normalized Mutual Information

com 16.257 páginas e CNET com 352.770 páginas. Nas outras duas coleções IG com 34.460 e BLOGS com 54.044 páginas os resultados ficaram muito próximos.

É interessante observar que a coleção na qual obtivemos maior ganho (CNET) é a coleção em que o *baseline* se mostrou instável e também é a coleção com maior quantidade de páginas entre as coleções usadas para os experimentos.

Na Tabela 4.8 são apresentadas as médias das métricas AdjRand e NMI calculadas sobre cada uma das 457.542 páginas Web de toda a coleção. Os valores apresentados nessa tabela nos permite verificar a superioridade dos resultados obtidos em nossos experimentos sobre toda a base de páginas que foram segmentadas.

Tabela 4.8: Valores médios de AdjRand e NMI sobre as 457.542 páginas de toda as coleções usadas nos experimentos.

		SVM	*Baseline*	Oráculo
Toda a Base	AdjRand	0,714	0,576	0,847
	NMI	0,830	0,763	0,908

Na Figura 4.4 podemos observar o comportamento de cada método através das métri-

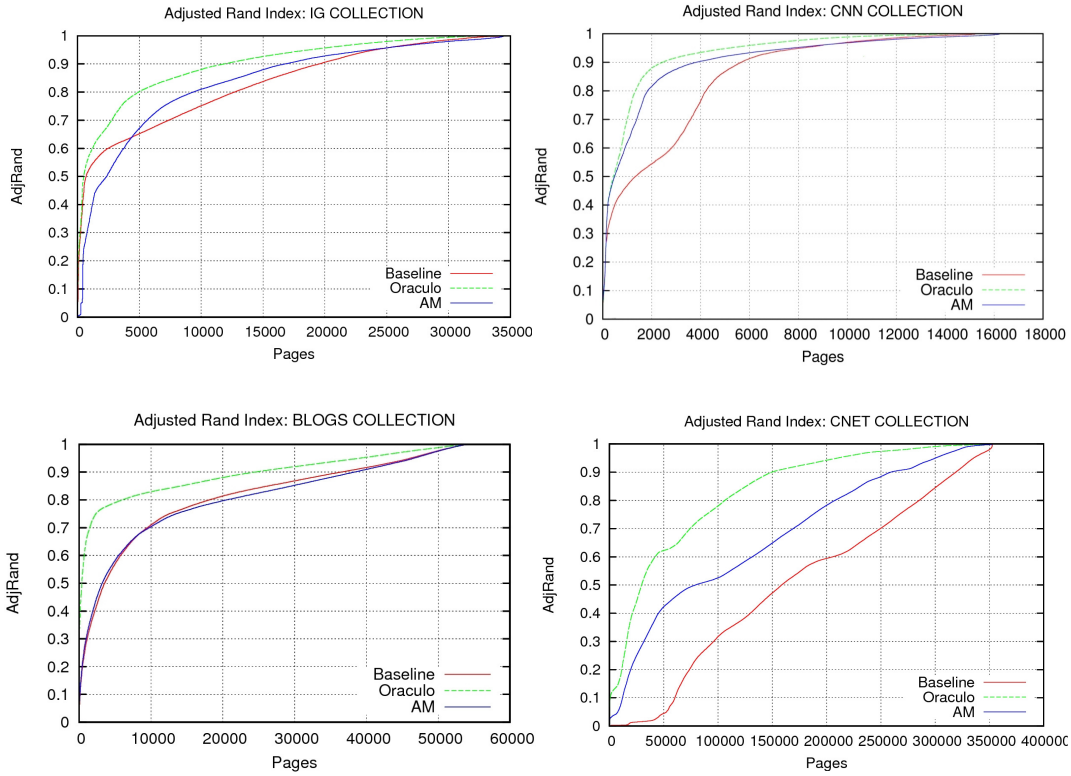

Figura 4.3: Avaliação das coleções IG, CNN, BLOGS e CNET com a metrica Adjusted Rand Index

cas AdjRand e NMI para todas as páginas usadas nos experimentos. Para cada métrica foi desenhado um gráfico utilizando as 457.542 páginas avaliadas. Cada método tem seu desempenho através de um curva no gráfico. Então, cada curva nos gráficos da figura denota o desempenho de um dos métodos através de uma das métricas para toda a base de páginas Web utilizada.

No gráfico presente na Figura 4.4 com as medidas de Adjusted Rand Index, podemos observar que aproximadamente 180.000 páginas obtiveram valores de Adjusted Rand Indez inferiores a 0,7 em nossos experimentos, enquanto que mais de 260.000 páginas tiveram Adjusted Rand Index a esse valor para o *baseline*. Para a métrica NMI o comportamento é semelhante. No segundo gráfico da mesma figura cerca de 40.000 páginas obtiveram valores de NMI a baixo de 0,7 para o método que utiliza técnicas de aprendizagem de máquina, enquanto quase 100.000 páginas estão a baixo de 0,7 no *baseline*.

Nas Tabelas 4.7, 4.8 e nas Figuras 4.2 4.3 e 4.4, são apresentadas as medidas e as curvas das métricas AdjRand e NMI obtidas com o classificador oráculo. Esses resultados mostram a maior proximidade com a segmentação manual que pode ser alcançada

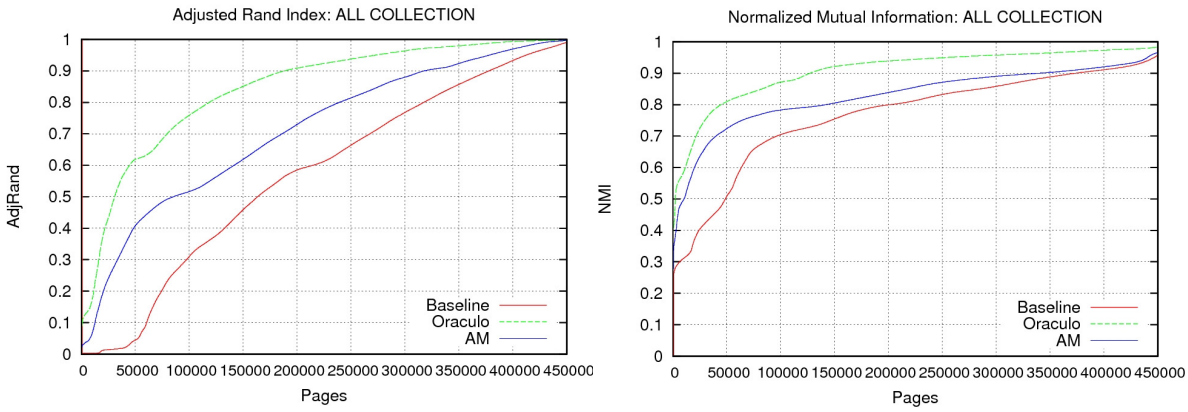

Figura 4.4: Gráfico com avaliação de toda a base de páginas Web com as métricas Normalized Mutual Information e Adjusted Rand Index

utilizando a estratégia de poda da árvore SOM tanto no *baseline* quanto nos nossos experimentos.

Há dois principais motivos para existir esse topo definido pelo oráculo, tais motivos estão relacionados ao primeiro passo do processo de segmentação estudado e à abordagem utilizada no terceiro passo. Como vimos no capítulo 3.3, o primeiro passo é um processo de poda sobre as árvores DOM em busca de área com estrutura regular e áreas com conteúdo aninhado.

O primeiro motivo esta relacionado à saída do primeiro passo quando um conjunto de nós pertencentes a blocos distintos são erroneamente combinados a um único bloco. Neste caso o erro de segmentação já esta na árvore DOM pré-processada antes desta ser adicionada a árvore SOM e por consequência, antes de qualquer processo de poda sobre a árvore SOM.

Para entendermos o segundo motivo vamos relembrar a abordagem adotada. Lembre-se que na árvore SOM o processo de poda se dá a partir da verificação de sucessivos grupos de nós formados por um nó pai e seus respectivos nós folhas, cada grupo é classificado em *B* (Elementos pertencentes a um único bloco) ou *NB* (Elementos pertencentes a blocos distintos). Quando um grupo é classificado como sendo da categoria *B* os nós filhos deste grupo são podados da árvore SOM e o conteúdo textual do grupo é associado ao nó pai. Quando classificados como elementos da categoria *NB* não ocorre poda e os nós folhas são mantidos como block class. Nessa abordagem assumimos que cada bloco esta associados a um único nó e este não pode conter outros blocos o que é verdade para uma grande maioria dos casos porém há exceções. Uma exceção ocorre quando parte

de um grupo de nós irmãos deve ser combinado em um bloco diferente dos demais nós irmãos. Neste caso, combinar todos os elementos ao nó pai (categoria *B*) leva a uma pequena falha na segmentação, assim como manter todos os irmãos separados (categoria *NB*).

4.4 Análise dos Resultados

O resultado obtido com este método de segmentação está limitado aos reportados com uso do oráculo. Sabe-se que os motivos que limitam o resultado obtido com o oráculo são os mesmo para qualquer outra função que siga este principio de combinar ou não todos os filhos de um nó em um único elemento. Para o método baseado em aprendizagem de máquina a influência é ainda maior, pois o processo de aprendizagem depende de exemplos (amostras rotuladas) que são obtidos com a classificação do oráculo. As limitações descritas para o oráculo impedem que consigamos amostras de uma segmentação ótima além de limitar a margem de ganho sobre o *baseline*. Lembre-se que a função gerada pelo processo de AM é baseada nas informações obtidas com o oráculo, logo uma função perfeita, que consiga obter acurácia de 1,0 será tão boa quando o próprio oráculo.

Capítulo 5

Conclusões e Trabalhos Futuros

Esse trabalho se propôs a demonstrar que é possível obter bons resultados em um método automático de segmentação de páginas Web utilizando técnicas de aprendizagem de máquina. Sendo mais específico, obter boa segmentação utilizando classificadores binários gerados por aprendizado supervisionado a partir de exemplos extraídos das páginas de outros Web sites, diferente do que está sendo segmentado.

As técnicas de aprendizagem de máquina foram utilizadas para aprimorar o método de segmentação automática de páginas Web, chamado SOM_{tree} (Árvore SOM). Esse método proposto em [11], é capaz de particionar páginas em blocos de acordo com a modelagem necessária para o sistema de ranking baseado em blocos proposto em [14, 11]. Em [11] foi demonstrado que os resultados do sistema de ranking baseado em blocos, utilizando a segmentação obtida através do método SOM_{tree} ficaram próximos aos resultados utilizando a segmentação de referência. Apesar desses resultados, para sua execução o processo de segmentação da árvore SOM utiliza duas características dos nós dessa estrutura para aplicar limiares (α e β) com valores fixo definidos empiricamente.

Em nosso trabalho, definimos uma maior quantidade de características para representar os nós da árvore SOM. A partir do *baseline* utilizamos uma estratégia com foco em identificar os nós que representam classe de blocos nessa estrutura. Para isso, extraímos exemplos (grupo de nós da árvore SOM) cuja informação de ser ou não uma classe de bloco é obtida de páginas segmentadas manualmente. Os exemplos extraídos formam bases de treino que são fornecidas ao método SVM para gerar classificadores automaticamente com base dos exemplos. Avaliamos esses classificadores, testando-os em bases extraídas de páginas de outros Web sites. Para demonstrar os bons resultados dos classifi-

cadores, os comparamos aos resultados obtidos com o classificador utilizado no *baseline* nas mesmas bases de teste.

Ao avaliarmos a segmentação obtida por meio de métricas de avaliação de algoritmos de *clustering*, comprovamos o bom resultado da segmentação com uso de técnicas de aprendizagem de máquina sobre a estrutura *árvore SOM*, comprovando também que é possível segmentar páginas a partir de exemplos extraídos de outros Web sites. Verificamos ainda que os resultados de segmentação obtidos se mostram superiores ao método SOM_{tree} original.

Um diferencial da segmentação automática utilizando a árvore SOM é que uma página não é segmentada isoladamente. As páginas HTML são relacionadas durante o processo de segmentação, levando a uma segmentação mais coesa. Isso porque páginas com estrutura semelhante tendem a possuir a mesma estrutura de blocos.

5.1 Trabalhos Futuros

A maioria dos métodos de segmentação automática encontrados na literatura utilizam informações apenas da própria página que esta sendo segmentada. Apesar do método que está sendo proposto utilizar informações de todo o site (obtidas a partir da árvore SOM), idéias similares a apresentada neste trabalho podem ser aplicadas também na árvore DOM. Para isso, é preciso apenas desconsiderar as evidências que só podem ser obtidas com a árvore SOM.

A avaliação dos resultados obtidos em nosso trabalho considerou sua aplicação em áreas que uma segmentação próxima a percepção que o usuário tem como ideal seja necessária. Para isso comparamos os resultados obtidos com nosso método automático aos resultados de parâmetros obtidos através de um processo de segmentação manual. No entanto uma importante aplicação de métodos de segmentação automática é sua utilização em algoritmos de busca que consideram a estrutura das páginas para melhorar seus resultados. Em especial o método SOM_{tree} particiona páginas em blocos de acordo com a modelagem necessária para o sistema de ranking proposto em [14]. Sendo assim é essencial que sejam realizados testes com esse método de ranking.

Apesar dos bons resultados alcançados em nosso trabalho, identificamos que com a abordagem utilizada no processo de segmentação da árvore SOM não é possível refinar-

mos o classificador de modo que seja possível chegar a uma segmentação idêntica a segmentação manual, considerada ótima. Sendo assim, é válido um estudo para identificar novas formas de podar a árvore SOM. Por exemplo, uma variação do nosso processo seria utilizar um classificador para decidir entre combinar ou não combinar pares de nós irmãos em pertencentes ou não ao mesmo bloco, assim um conjunto de todos os nós irmãos só seria combinado após cada nó ser classificado como pertencente ao mesmo bloco. Essa estratégia permitiria que parte de um conjunto de nós irmãos fossem combinados em um bloco diferente dos demais.

Referências Bibliográficas

[1] ALCIC, S., AND CONRAD, S. Page segmentation by web content clustering. In *Proceedings of the International Conference on Web Intelligence, Mining and Semantics* (2011), ACM, p. 24.

[2] ALGUR, S. P., AND HIREMATH, P. Extraction of flat and nested data records from web pages. In *Conferences in Research and Practice in Information Technology Series* (2006), vol. 245, pp. 163–168.

[3] ALPAYDIN, E. *Introduction to machine learning*. MIT press, 2010.

[4] ÁLVAREZ, M., PAN, A., RAPOSO, J., BELLAS, F., AND CACHEDA, F. Extracting lists of data records from semi-structured web pages. *Data & Knowledge Engineering 64*, 2 (2008), 491–509.

[5] BURGES, C. J. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery 2*, 2 (1998), 121–167.

[6] CASTRO, J., FLORES-HIDALGO, L., MANTAS, C., AND PUCHE, J. Extraction of fuzzy rules from support vector machines. *Fuzzy Sets and Systems 158*, 18 (2007), 2057–2077.

[7] CHAKRABARTI, D., KUMAR, R., AND PUNERA, K. A graph-theoretic approach to webpage segmentation. In *Proceeding of the 17th international conference on World Wide Web* (2008), ACM, pp. 377–386.

[8] CHANG, C.-C., AND LIN, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology 2* (2011), 27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[9] CHAPELLE, O., VAPNIK, V., BOUSQUET, O., AND MUKHERJEE, S. Choosing multiple parameters for support vector machines. *Machine learning 46*, 1 (2002), 131–159.

[10] D. CAI, S. YU, J.-R. W., AND MA., W.-Y. Vips: a vision-based page segmentation algorithm. Tech. rep., Microsoft Technical Report, 2003.

[11] DE MOURA, E., FERNANDES, D., RIBEIRO-NETO, B., DA SILVA, A., AND GONÇALVES, M. Using structural information to improve search in web collections. *Journal of the American Society for Information Science and Technology 61*, 12 (2010), 2503–2513.

[12] DUAN, K., KEERTHI, S. S., AND POO, A. N. Evaluation of simple performance measures for tuning svm hyperparameters. *Neurocomputing 51* (2003), 41–59.

[13] FERNANDES, D., DE MOURA, E., DA SILVA, A., RIBEIRO-NETO, B., AND BRAGA, E. A site oriented method for segmenting web pages. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval* (2011), ACM, pp. 215–224.

[14] FERNANDES, D., DE MOURA, E., RIBEIRO-NETO, B., DA SILVA, A., AND GONÇALVES, M. Computing block importance for searching on web sites. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management* (2007), ACM, pp. 165–174.

[15] FRASCONI, P., AND SHAMIR, R. *Artificial intelligence and heuristic methods in bioinformatics*, vol. 183. Ios PressInc, 2003.

[16] FU, X., ONG, C., KEERTHI, S., HUNG, G. G., AND GOH, L. Extracting the knowledge embedded in support vector machines. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on* (2004), vol. 1, IEEE.

[17] HATTORI, G., HOASHI, K., MATSUMOTO, K., AND SUGAYA, F. Robust web page segmentation for mobile terminal using content-distances and page layout information. In *Proceedings of the 16th international conference on World Wide Web* (2007), ACM, pp. 361–370.

[18] IMBAULT, F., AND LEBART, K. A stochastic optimization approach for parameter tuning of support vector machines. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on* (2004), vol. 4, IEEE, pp. 597–600.

[19] JOACHIMS, T. *Learning to classify text using support vector machines: Methods, theory and algorithms*, vol. 186. Kluwer Academic Publishers Norwell, MA, USA:, 2002.

[20] KIM, K. I., JUNG, K., PARK, S. H., AND KIM, H. J. Support vector machines for texture classification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 24*, 11 (2002), 1542–1550.

[21] KOHLSCHÜTTER, C., AND NEJDL, W. A densitometric approach to web page segmentation. In *Proceeding of the 17th ACM conference on Information and knowledge management* (2008), ACM, pp. 1173–1182.

[22] LIU, B., GROSSMAN, R., AND ZHAI, Y. Mining data records in web pages. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (2003), ACM, pp. 601–606.

[23] LORENA, A. C., AND DE CARVALHO, A. C. Uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada 14*, 2 (2007), 43–67.

[24] MEHTA, R. R., MITRA, P., AND KARNICK, H. Extracting semantic structure of web documents using content and visual information. In *Special interest tracks and posters of the 14th international conference on World Wide Web* (2005), ACM, pp. 928–929.

[25] PONTIL, M., AND VERRI, A. Support vector machines for 3d object recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 20*, 6 (1998), 637–646.

[26] RICH, E., KNIGHT, K., CALERO, P. A. G., AND BODEGA, F. T. *Inteligência artificial*, vol. 2. McGraw-Hill, 1994.

[27] STREHL, A., AND GHOSH, J. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research 3* (2003), 583–617.

[28] VAPNIK, V. *The nature of statistical learning theory*. springer, 1999.

[29] WARRENS, M. J. On the equivalence of cohen's kappa and the hubert-arabie adjusted rand index. *Journal of Classification 25*, 2 (2008), 177–183.

[30] WEISS, S., AND KULIKOWSKI, C. Computer systems that learn.

[31] YI, L., LIU, B., AND LI, X. Eliminating noisy information in web pages for data mining. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (2003), ACM, pp. 296–305.