

2 Pesquisa Bibliográfica

As estratégias para detecção de conteúdo relevante podem ser divididas em duas abordagens: orientada a *site* e orientada a página. A primeira técnica utiliza várias páginas de um mesmo *site* para detectar *templates*. As desvantagens desta técnica é que nem todos *sites* na internet mantêm o mesmo modelo de estilo em suas páginas e se um novo *site* surge o modelo criado pode não ser mais aplicável. A vantagem deste método é que determinadas categorias de *sites* são bem regulares, como por exemplo *sites* de comércio virtual.

A segunda técnica procura *templates* por página. Esta técnica usualmente examina apenas as características dos nós da árvore DOM sem comparar com outras árvores DOM. A desvantagem desta técnica é que muitas vezes as características do nó DOM é vista isoladamente, o que pode diminuir a acurácia do método em um *site* regular. A vantagem é poder generalizar o método para diferentes categorias de *sites*.

Este capítulo discute alguns trabalhos que utilizaram a técnica orientada a *site* na Seção 2.1 e outros que utilizaram a técnica orientada a página na Seção 2.2.

2.1 Estratégias orientadas a site

2.1.1 Uma técnica de extração de templates baseada no estilo de apresentação

Yi et. al em (YLL03) classifica um conjunto de páginas com descrição de produtos e críticas de produtos em diferentes categorias. O objetivo de Yi é mostrar que *templates* nas páginas podem impactar negativamente o desempenho de tarefas de mineração de dados, como classificação em categorias e agrupamento.

A idéia descrita em (YLL03) se baseia na observação dos autores que em um típico *site* comercial as páginas tendem a seguir um estilo de apresentação fixo, pois a maioria das páginas são geradas automaticamente. As partes da página cujo estilo e o conteúdo real, isto é, *links*, texto, imagens, dentre outros, aparecem repetidamente em muitas páginas do *site* é mais provável que seja

um *template*. O conteúdo da página são as partes da página cujo conteúdo real e estilo não se repetem em outras páginas.

A partir desta observação Yi propõe uma estratégia que constrói uma nova estrutura de dados baseada no estilo da página a partir de uma árvore DOM, esta nova estrutura também é uma árvore e é chamada de SST (*Site Style Tree*). O estilo que os autores consideram são cor de fundo, cor da fonte, tamanho da fonte, dentre outros atributos que um CSS¹ pode conter. Uma análise do conteúdo real também faz parte da estratégia criada.

A construção de uma SST começa com o parser do conjunto de páginas do *site*, esta etapa produz um conjunto de árvores DOM. Estas árvores são mapeadas para uma árvore SST. Veja um exemplo de mapeamento na Figura 2.1. Os nós com o mesmo estilo que possuem o mesmo pai são agrupados em um nó na SST, veja que os três nós <td> se tornaram apenas um. Existe um contador para cada nó da SST representando o número de árvores DOM que aquele nó se encontra, por exemplo observe que o nó <p> na SST apresenta o número um no seu canto superior esquerdo porque ele só existe na árvore DOM da direita.

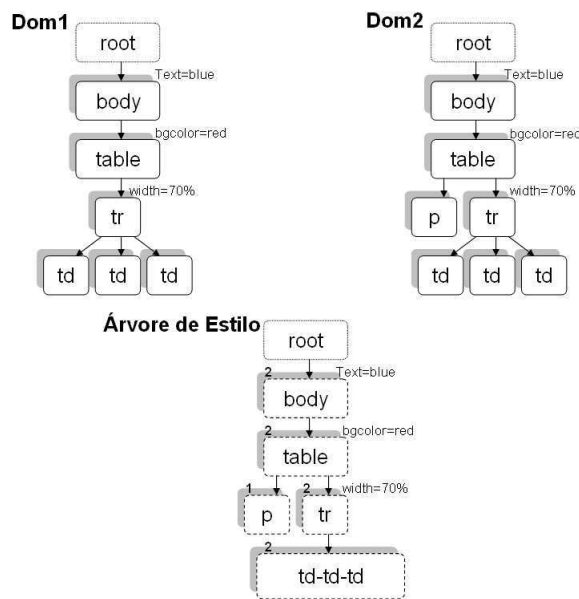


Figura 2.1: Exemplo de árvore DOM para ST

Com o término da construção da SST o próximo passo é identificar a importância de um nó da SST atribuindo uma pontuação ao nó. Esta pontuação pode ser calculada a partir das características do seu estilo de apresentação e do seu conteúdo real, para mais detalhes deste cálculo consulte

¹<http://www.w3.org/Style/CSS/>

(YLL03). Após esta atribuição, as páginas do *site* podem ter seus *templates* extraídos utilizando a pontuação calculada para cada nó da SST construída.

2.1.2

Uma técnica de extração de templates baseada em distância de edição entre árvores

Vieira et. al (VdSP⁺06) propõe outro algoritmo que adota a estratégia orientada a *site*. Neste trabalho o objetivo é melhorar o desempenho das tarefas de classificação e agrupamento em *sites* comerciais removendo *templates* das páginas. Metade dos sites comerciais considerados no corpus são *sites* de comércio eletrônico e a outra metade são *sites* de diferentes domínios, como CNN, E-Jazz, Encyclopedia Mythica, UBL (Ultimate Band List) e Wikipedia.

A motivação do algoritmo desenvolvido por Vieira, chamado de RTDM-TD, veio da observação que *templates* são apenas fragmentos de código HTML dentro de uma coleção de documentos HTML em regiões específicas da página. Isto acontece porque *sites* comerciais normalmente utilizam ferramentas que geram código HTML automaticamente. A idéia dos autores é, intuitivamente, identificar uma subárvore que se repete ao longo de páginas de um *site*, esta subárvore então é considerada o *template* do *site*.

Para obter o *template* de um conjunto de páginas Vieira et. al determina o mapeamento entre a estrutura das árvores. Este mapeamento detecta nós idênticos nas árvores e subárvores que contém estes nós. Após a identificação deste mapeamento, em um pequeno conjunto de páginas do *site*, as outras páginas do *site* podem ter seus *templates* extraídos. O funcionamento do algoritmo de mapeamento pode ser consultado em (VdSP⁺06).

A estratégia proposta por Vieira reduz o problema de detecção de *templates* a achar a subárvore em comum a um conjunto de árvores. O primeiro passo da estratégia é construir um conjunto das árvores DOM das páginas de um *site* através de um parser. A seguir duas árvores T_0 e T_1 são selecionadas aleatoriamente e removidas deste conjunto árvores. O segundo passo extrai a subárvore s_{i-1} em comum entre T_0 e T_1 . O próximo passo consiste em um laço onde uma árvore T_i é selecionada aleatoriamente na i -ésima iteração. O algoritmo verifica se T_i contém a subárvore s_{i-1} , se s_{i-1} não está contida em T_i uma outra subárvore s_i é extraída entre s_{i-1} e T_i , caso contrário o procedimento segue para a próxima iteração. Quando o número de iterações terminar a última subárvore s_i é retornada. O número de iterações é o número mínimo de páginas que precisa ser examinada para se obter o *template* mais geral possível.

2.1.3

Uma técnica de extração de templates baseada em pagelets

Bar-Yossef e Rajagopalan apresentam em (BYR02) dois algoritmos que também utilizam várias páginas de um *site* para detectar *templates*. Seu objetivo, assim como dos outros trabalhos descritos acima, é aplicar seus algoritmos a tarefas de mineração de dados.

A fim de detectar os *templates* Bar-Yossef e Rajagopalan definem o conceito de *pagelet*. Citando a definição semântica de *pagelet* encontrada no artigo:

Um pagelet é a região de uma página web que: 1) tem um tópico ou funcionalidade bem definidos; e 2) não está aninhado dentro de nenhuma outra região que tem o mesmo tópico ou funcionalidade.

O primeiro passo da estratégia é dividir a página em *pagelets*. Para atingir esta finalidade primeiro é feito o parser de uma página. Uma pilha armazena a raiz como primeiro elemento. O próximo passo é iterar em um laço que termina quando a pilha esta vazia. Dentro do laço uma variável v recebe um elemento que é desempilhado e então é verificado se os descendentes de v tem ao menos k links, se isto acontece os descendentes de v são empilhados, caso contrário v é declarado como um *pagelet*. Este algoritmo de particionamento é baseado na idéia de que se um elemento HTML contém pelo menos um determinado número de links então é provável que represente um tópico ou idéia independente.

Após a divisão em *pagelets* um dos dois algoritmos propostos podem ser executados. O algoritmo chamado de **Local Template Detection** é adequado para um conjunto pequeno de páginas enquanto o algoritmo chamado **Global Template Detection** é adequado para um conjunto grande de páginas.

No algoritmo **Local Template Detection** a primeira etapa elimina as páginas duplicadas do seu domínio de páginas. A segunda etapa ordena e agrupa os *pagelets* de todo seu universo de páginas de acordo com seu *shingle*. Um *shingle* é uma representação única de um texto que não varia se existe pequenas perturbações. Nesta segunda etapa os grupos formados são *pagelets* candidatos a *templates*. A terceira etapa enumera os grupos e retorna os *pagelets* pertencentes a cada grupo.

O algoritmo **Global Template Detection** é mais sofisticado. O primeiro passo monta o conjunto dos *shingles* dos *pagelets* que ocorrem pelo menos duas vezes no conjunto de *pagelets* construído, aquele conjunto é chamado de T_s . O segundo passo é extrair os *pagelets* que ocorrem no conjunto T_s , o conjunto destes *pagelets* é chamado T_c . A seguir, para cada *shingle* $s \in T_s$, o algoritmo seleciona as páginas que contêm s . O conjunto destas páginas é chamado de G_s . O próximo passo é achar os links entre as páginas de G_s ,

para cada s , estes *links* formam o conjunto T_l . O próximo passo enumera os *shingles* em T_s e para cada um carrega na memória todos os *links* entre páginas de G_s . Finalmente, é utilizado um algoritmo BFS para obter as componentes conexas e não-direcionadas em G_s . Cada componente é um *template* ou não. As componentes que são *templates* são retornadas como resultado.

2.2

Estratégias orientadas a páginas

2.2.1

Uma técnica de extração de conteúdo relevante baseada em tags

Lin e Ho em (LH02) apresentam um algoritmo para detectar conteúdo informativo em páginas de *sites* de notícias utilizando a estratégia orientada a página. Os autores também propõem que este algoritmo utilize a *tag* <TABLE> para particionar a página e separar o conteúdo da página em conteúdo informativo e conteúdo não informativo. Algumas *tags*, em particular, também são consideradas, como `title`, `headings`, `<p>`, `<tr>` e `<td>`.

O algoritmo proposto em (LH02) se baseia na idéia que o conteúdo de uma página é estrutura de em tabelas. Portanto a idéia é localizar qual tabela possui o conteúdo desejado.

O algoritmo proposto tem cinco fases. A primeira fase extrai blocos de conteúdo de uma página. Blocos de conteúdo fazem parte de uma estrutura primitiva de árvore, a qual é obtida por um *parsing* da página HTML baseado na *tag* <TABLE>. Cada nó interno desta estrutura indica um bloco de conteúdo que consiste de uma *string*, ou mais, do conteúdo como suas folhas. A segunda etapa extrai atributos dos blocos de conteúdo. Neste caso os atributos são as palavras significativas. As *stop words* não são incluídas. A terceira etapa calcula o valor da entropia dos atributos. O valor da entropia de um atributo é estimado de acordo com o peso da distribuição dos atributos aparecendo em um conjunto de páginas de um *site*. A quarta etapa estima a entropia dos blocos de conteúdo. A entropia de um bloco de conteúdo, chamada de $H(CB)$, é a média de todas as entropias dos atributos do bloco. A última etapa classifica os blocos em informativo e não informativos. Se um bloco possui uma entropia $H(CB)$ menor que um limiar então ele é informativo, caso contrário ele é não informativo. O limiar é determinado por uma estratégia gulosa, aplicada para um conjunto de páginas.

2.2.2

Uma técnica de localização de conteúdo baseada em segmentação de página HTML

Deng et. al descrevem um algoritmo, chamado de VIPS, em (CYWM03) para localização de conteúdo através da segmentação da página em segmentos ou blocos. Um bloco é definido por Deng como uma estrutura semântica de uma página *web*. Neste algoritmo a árvore DOM é transformada em uma nova estrutura chamada de *vision-based content structure* (VCS). Esta estrutura também está na forma de árvore, onde cada nó é um bloco. Cada bloco consiste de um conjunto de nós DOM.

A idéia é dividir a página em partes cujo conteúdo estejam relacionados. A partição que possui o conteúdo relevante poderá ser então detectada.

O processo que constrói um VCS tem três etapas principais: extração de blocos, detecção de separadores e a construção de uma estrutura de conteúdo. A extração dos blocos se inicia com o *parsing* das páginas. Em seguida cada nó DOM passa por uma avaliação, baseada em pistas visuais, para saber se ele forma um único bloco ou não. Se o nó DOM não forma um bloco então seus descendentes são processados da mesma maneira.

Cada bloco extraído recebe uma pontuação chamada de DoC (*Degree of Coherence*) que mede o grau de coerência do bloco. Quando todos os blocos são extraídos na página eles são armazenados em um conjunto. Os separadores entre estes blocos são identificados e um peso é atribuído para cada separador baseado nas propriedades dos seus blocos vizinhos. A hierarquia do estilo de apresentação é construída baseada nestes separadores. Após a construção da hierarquia do estilo, o procedimento verifica se as folhas da VCS atendem ou não ao grau de granularidade requerida. Se não atendem ao grau de granularidade o procedimento trata cada folha como uma página e a segmenta. Após todos estes passos a VCS montada é retornada como saída.

2.2.3

Uma técnica de extração de templates baseado em aprendizado de máquina e suavização isotônica

Chakrabarti et. al descrevem em (CKP07) um ambiente que foi construído com o intuito de extrair *templates*. O ambiente pode ser dividido em três passos: geração automática de dados de treino, classificação e suavização isotônica.

A geração automática de dados de treinamento é feita utilizando a estratégia orientada a *site*. Para cada *site*, o algoritmo obtém aleatoriamente um conjunto de páginas P . Então para cada página $p \in P$ e para cada nó

DOM $n \in p$, o hash $h(n)$ do nó n é calculado. Em seguida o algoritmo constrói um conjunto de nós I_α^+ que ocorrem em uma fração α de páginas em P .

Para a etapa da classificação inicialmente os atributos dos nós do conjunto I_α^+ são identificadas no contexto das páginas que elas aparecem. Um classificador é treinado usando os atributos escolhidos e tratando os nós do conjunto I_α^+ como nós da classe *templates*, sabendo que existe a classe *template* e a classe não-*template*.

Na última etapa os autores utilizaram um classificador de regressão logística (Mit97) que atribui uma probabilidade de cada nó ser *template*. A seguir uma suavização é aplicada na probabilidade de cada nó a fim de que a seguinte propriedade seja seguida,

Um nó em uma árvore é um template se e somente se todos os seus filhos são templates.

Após a suavização os *templates* podem ser enfim detectados.

2.2.4

Uma técnica de extração de conteúdo relevante baseada em aprendizado de máquina e atributos visuais

Outro algoritmo que utilizou a estratégia orientada a página foi implementado por Zheng et. al (ZSW07). Nesta implementação os autores propõem uma estrutura de dados diferente de uma árvore DOM para capturar o conteúdo de páginas de notícias. Esta estrutura também possui a organização de uma árvore. Cada nó nesta estrutura, chamado de bloco visual, é um nó DOM cuja área é maior que zero ao ser renderizado em um navegador. Um bloco visual b_1 é pai de outro bloco b_2 caso não haja outro bloco, contido em b_1 , que contém b_2 .

Após a construção da estrutura visual cada bloco tem seus atributos extraídos para treino ou teste. Os atributos extraídos são listados na Tabela 2.1.

Veja na Figura 2.2 a seção de uma página de notícia. O nó que engloba o corpo na notícia é a *tag* ``. Segue os valores dos atributos deste nó na estrutura de dados proposta em (ZSW07).

- Posição: $x = 155$, $y = 240$ e nível de aninhamento é 18.
- Formato da fonte: Tamanho da fonte é 13, não é negrito e não é itálico.
- Estatísticas: 0 imagens, 0 *links*, 1603 tamanho do texto, 0 parágrafos, 0 parágrafos itálicos, 3 parágrafos negritos e 0 tabelas.

Tabela 2.1: Atributos utilizados no algoritmo V-Wrapper

Nome	Descrição
X	Posição em relação ao eixo das abscissas, localizado no eixo superior do navegador.
Y	Posição em relação ao eixo das ordenadas, localizado no eixo esquerdo do navegador.
Nível de aninhamento	Quantos antecessores o bloco possui.
Largura	Largura do bloco.
Altura	Altura do bloco.
Tamanho da fonte	Tamanho da fonte do texto contido no bloco.
Fonte em negrito	Atributo booleano indicando se o texto do bloco é negrito.
Fonte em itálico	Atributo booleano indicando se o texto do bloco é itálico.
Número de Imagens	Número de imagens existentes na subárvore do bloco.
Número de <i>Links</i>	Número de <i>links</i> existentes na subárvore do bloco.
Número de Parágrafos	Número de <i>tags</i> <p> existentes na subárvore do bloco.
Número de Parágrafos Negritos	Número de <i>tags</i> existentes na subárvore do bloco.
Número de Parágrafos Itálicos	Número de <i>tags</i> <i> existentes na subárvore do bloco.
Número de Tabelas	Número de <i>tags</i> <table> existentes na subárvore do bloco.
Tamanho do texto	Número de caracteres do texto contido na subárvore do bloco. Se for uma folha, o número de caracteres do texto contido nela.
Largura relativa	A razão do tamanho da largura do bloco pela largura do pai.
Posição relativa	Posição do bloco em relação ao eixo das ordenadas menos a posição do pai em relação ao eixo das ordenadas.

Jean Azevedo é o 8º no primeiro dia em Dubai
29/10/07 - 14h18

Começou oficialmente hoje (29) o Rally de Dubai (JAE Desert Challenge), nos Emirados Árabes, válido como etapa do Campeonato Mundial. O brasileiro Jean Azevedo, que corre nas motos, marcou o 8º melhor tempo na etapa que teve 444 quilômetros.



A prova foi vencida por Marc Coma, seguido de Cyril Despres, em segundo, e Jordi Viladoms, em terceiro. "Foi uma etapa com trechos rápidos e tínhamos que estar muito atentos ao roadbook para evitar perigos. Está realmente muito quente por aqui, já vi alguns pilotos reclamando disso", comentou Jean, que largou na 16ª colocação.

O brasileiro estréia sua nova moto, uma KTM 690, que usará no Rally Dakar 2008. Ele teve pouco tempo para fazer os testes e ajustes necessários no novo equipamento. "Estou tentando trocar a suspensão da moto mas ainda não consegui. O problema é que em altas velocidades a motocicleta fica instável, dificultando a pilotagem. Precisamos resolver esse detalhe para poder testar também o segundo motor que levamos", comentou.

Nos carros, Nasser Al Attiyah/ Tina Toerner fizeram o melhor tempo. Luc Alphand/ Gilles Picard ficaram em segundo e Stéphane Peterhansel/ Jean Paul Cottret, em terceiro. Nesta terça-feira (30), os pilotos terão pela frente cerca de 382 quilômetros de percurso.

Resultados da primeira etapa:

Motos

- 1) Marc Coma (KTM) – 3h03min47s
- 2) Cyril Despres (KTM) – 3h06min18s
- 3) Jordi Viladoms (KTM) – 3h07min41s
- 8) Jean Azevedo (KTM) – 3h23min13s

Carros

- 1) Nasser Al Attiyah/ Tina Toerner (BMW) – 2h41min11s
- 2) Luc Alphand/ Gilles Picard (Mitsubishi) – 2h41min34s
- 3) Stéphane Peterhansel/ Jean Paul Cottret (Mitsubishi) – 2h42min05s

Figura 2.2: Exemplo dos atributos de blocos

- Estendidas: 1.0 é a razão da largura em relação ao pai e 0 é o deslocamento em relação ao pai considerando o eixo das ordenadas.

Este nó é pai dos nós que contém o texto do corpo da notícia. Porém veja que alguns valores não são acumulativos, por exemplo se o texto do nó é negrito. Esses valores são inicializados nas folhas. Um exemplo é o nó folha **Resultados da primeira etapa**, da Figura 2.2, que é negrito.

Após a extração dos atributos acontece a fase de rotulação do conjunto de treino. Nesta fase os blocos que são folhas são rotulados de acordo com as seguintes classes: título, conteúdo e outros. Um nó pertence à primeira classe quando o texto nele contido é o título da notícia, um nó pertence à segunda classe quando o seu texto está contido no conteúdo da notícia e o último rótulo se aplica a nós cujo conteúdo não faz parte do título da notícia nem do conteúdo da notícia. Após os nós folhas serem rotulados, os nós internos são rotulados com as seguintes classes: Positivo e Negativo. Um nó é da classe Positivo quando pelo menos um de seus filhos é da classe título, conteúdo ou Positivo. Um nó é da classe Negativo quando todos seus filhos são da classe Negativo ou outros.

Depois da rotulação do conjunto de dados os nós internos são separados em um conjunto B_1 e um classificador é treinado neste conjunto. Neste mesmo conjunto B_1 o classificador que foi treinado é aplicado. As folhas, cujo pai está em B_1 e foi classificado como Positivo, são separadas para construir um conjunto B_2 . Um classificador é treinado neste conjunto B_2 . Portanto a estratégia é constituída de dois classificadores, um para nós internos e outro para nós folhas.

Finalmente a última fase constitui-se da extração do conteúdo. Nesta fase o primeiro passo é classificar os nós internos do conjunto de dados. O algoritmo então percorre a estrutura de blocos procurando por folhas cujos pais foram classificados como Positivo. A busca para quando encontra uma folha, neste caso a folha é retornada como conteúdo, ou quando encontra um nó Negativo, neste caso não existe conteúdo para retornar. O segundo passo utiliza o classificador de folhas para rotular as classes das folhas selecionadas no primeiro passo. O resultado do segundo passo indica qual é o conteúdo a ser extraído da página de notícia.