3
Abordagem

O trabalho (Liu et. al., 2003) foi utilizado como base para as abordagens apresentadas neste capítulo. Em seu trabalho, Liu *et. al.* apresentam um procedimento de comparação de árvores para a identificação de listagens, ou, como definido pelos autores, *data regions*. Um fato curioso ao ser observado nesse trabalho é que a árvore DOM não foi utilizada diretamente, mas sim foi criada uma *árvore de etiquetas* (Tag Tree), que tem estrutura muito semelhante à DOM. A heurística utiliza o conhecimento da ordenação da árvore DOM, que é mantida pela Tag Tree, para detectar sub-árvores que apresentam a mesma estrutura.

Neste capítulo, são propostos procedimentos para o cálculo de distância entre duas árvores e para a busca do conjunto de nós que devem ser comparados, a fim de identificar as sub-estruturas semelhantes na árvore de um documento HTML. Para apresentá-las, são utilizadas algumas das definições propostas em (Liu et. al., 2003). Além disso, também é descrito o ambiente que foi desenvolvido para tornar possível a realização de experimentos e avaliar a qualidade dos algoritmos desenvolvidos nesta dissertação.

PUC-Rio - Certificação Digital Nº 0821371/CA

3.1
Algoritmos de similaridade em árvore

Em uma árvore DOM, podem existir sub-árvores semelhantes, que definem uma região visual chamada de *data region*. Para encontrar uma *data region* em um documento HTML, Liu *et. al.* (Liu et. al., 2003) definem os *conjuntos generalizadores* por serem estruturas mais simples e, com isso, mais fáceis de identificar do que as *data regions*. Os procedimentos que identificam os conjuntos generalizadores são chamados de **funções de busca** e são apresentados na Seção 3.1.2.

Definição 1 Conjunto Generalizador. *Um conjunto generalizador de tamanho r consiste em $r \geq 1$ nós DOM com as seguintes propriedades:*

1. *todos os nós são filhos do mesmo pai;*
2. *todos os nós são consecutivos, assumindo uma ordenação da esquerda para a direita.*

Um conjunto generalizador nada mais é que um conjunto de nós da árvore DOM que respeitam algumas condições. Essa definição é necessária, pois facilita a formalização do problema e também ajuda na Definição 2. Além disso, a razão por ser introduzido o conjunto generalizador é que esse captura a situação onde um registro é descrito por mais de um nó adjacente. Por exemplo, na Figura 3.1 pode ser observado que cada registro é apresentado por cinco linhas de uma tabela (tags tr).

Definição 2 DATA REGION. *Uma data region é uma coleção com dois ou mais conjuntos generalizadores com as seguintes propriedades:*

1. *todos os conjuntos generalizadores são filhos do mesmo pai;*
2. *todos os conjuntos generalizadores têm o mesmo tamanho;*
3. *todos os conjuntos generalizadores são consecutivos;*
4. *A distância normalizada entre os conjuntos generalizadores consecutivos é menor que um dado* $\alpha \geq 0$.

PUC-Rio - Certificação Digital Nº 0821371/CA

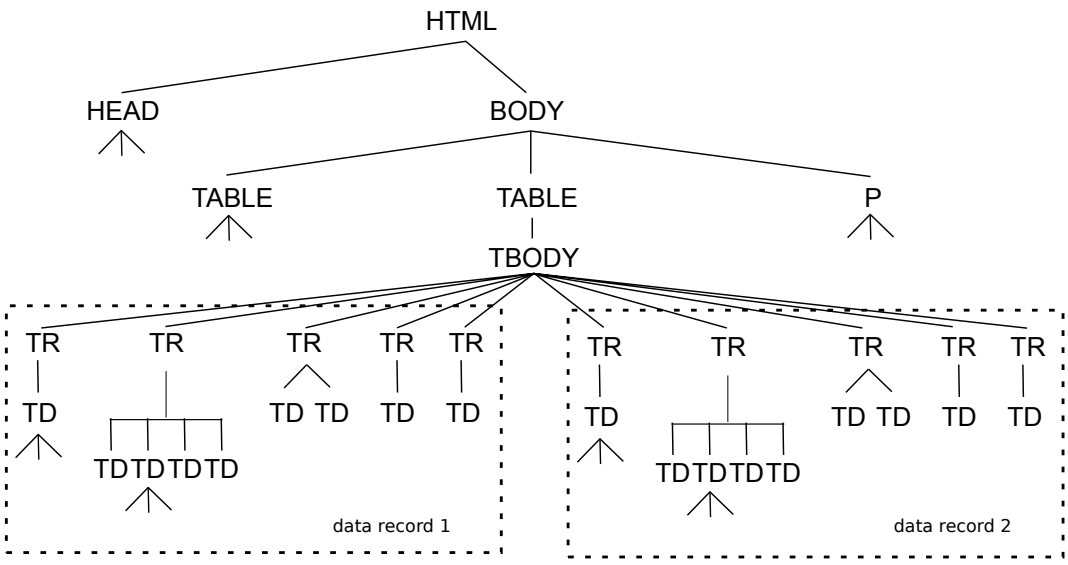

Figura 3.1: Exemplo de dois conjuntos generalizadores de tamanho 5

Por exemplo, na Figura 3.1 são apresentados dois conjuntos generalizadores. O primeiro consiste das primeiras cinco tags tr. O segundo consiste nas outras cinco tags tr subsequentes. É importante relembrar que os conjuntos generalizadores de um *data region* têm o mesmo tamanho, e mesmo se as sub-estruturas dos conjuntos generalizadores forem um pouco diferentes essas podem ser consideradas semelhantes, caso a distância entre elas seja menor que α.

Para tornar mais claro os diferentes tipos de agrupamentos que podem criar um conjunto generalizador, é apresentada uma árvore artificial na Figura 3.2, onde os nós dos conjuntos generalizadores estão em cinza. Para convencionar a notação, é utilizado um número para identificar o nó, no lugar do nome da tag. Os nós 5 e 6 são dois conjuntos generalizadores de tamanho 1 e juntos definem a *data region* 1. Os nós 8, 9 e 10 também são conjuntos generalizadores de tamanho 1 e juntos definem a *data region* 2, sempre observada a Condição 4 da Definição 2. Os pares de nós (14,15) e (16,17) são conjuntos generalizadores com tamanho 2, e definem a *data region* 3.

Mesmo utilizando as definições apresentadas em (Liu et. al., 2003), as abordagens aqui propostas utilizam duas alterações importantes. A primeira é a utilização da árvore DOM, em vez da Tag Tree apresentada pelos autores. Como a árvore DOM é uma estrutura definida pela W3C, como visto no Capítulo 2, acreditamos que essa é uma estrutura madura que reflete o documento HTML com precisão. A segunda é a aplicação de um pós processamento no conjunto retornado para que só existam listas ou tabelas dos tipos desejados. Essa segunda adaptação será explicada durante a descrição da abordagem de cada tarefa.

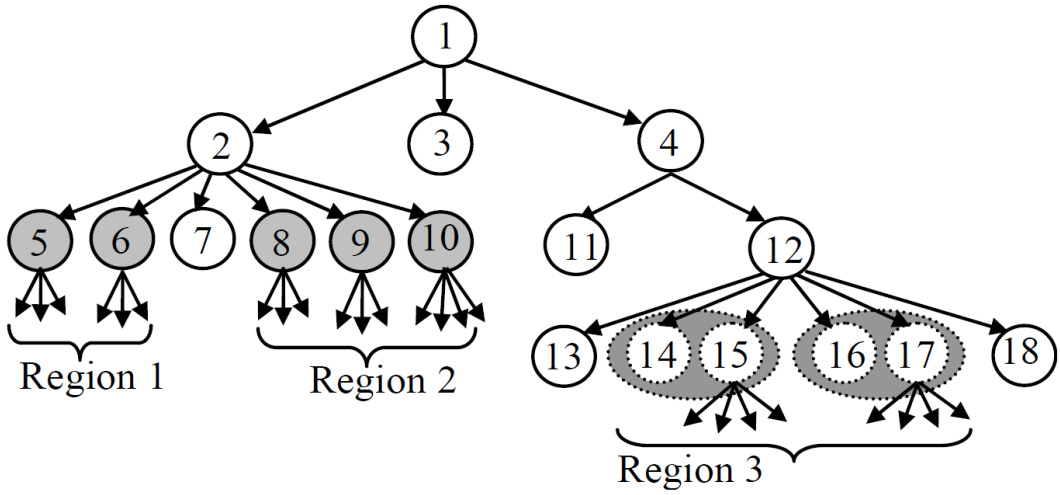

Figura 3.2: Exemplo de *data regions* (Regions), retirado de (Liu et. al., 2003)

Tendo em vista as definições descritas, podem ser observados dois pontos importantes onde a abordagem pode apresentar grande influência nos resultados. A forma com que os conjuntos generalizadores são comparados é um ponto crucial para a detecção dos *data regions*, pois essa é a principal regra para a criação dessas estruturas (Condição 4 da Definição 2). Por esse motivo, a Seção 3.1.1 é direcionada à descrição dos procedimentos que calculam a distância entre os conjuntos generalizadores. O segundo ponto importante a ser observado

PUC-Rio - Certificação Digital Nº 0821371/CA

é a criação dos conjuntos generalizadores. Sendo assim, a Seção 3.1.2 apresenta algumas alternativas para a criação dos conjuntos generalizadores.

3.1.1 Funções de distância

Esta seção apresenta três procedimentos para o cálculo das distâncias que são utilizados para garantir a Condição 4 da Definição 2. Primeiramente, são descritas duas adaptações da função de Levenshtein (Gusfield, 1997), tornando possível sua aplicação em árvores. Em seguida, é descrito o procedimento proposto em (Yang, 1991), utilizado em (Zhai et. al., 2005), que também é replicado nesta dissertação. Existem diversas outras técnicas para o cálculo de distâncias em árvore ou em outros tipos de estruturas que poderiam ser adaptadas e utilizadas, porém os três procedimentos aqui apresentados são simples e proporcionaram resultados interessantes como será discutido nos Capítulos 4 e 5.

A função de distância de Levenshtein é conhecida e comumente utilizada no universo de comparação de sequências de caracteres. Por esse motivo, é comum sua adaptação a problemas para que seja possível utilizá-la como função de distância. Um exemplo de adaptação da função de Levenshtein é encontrado na comparação de duas árvores (Liu et. al., 2003) feita utilizando uma sequência de caracteres criada a partir da concatenação dos nomes dos nós em uma navegação em profundidade (DFS). Esse é o primeiro dos três procedimentos utilizados para o cálculo da distância entre árvores.

Distância em Caracteres (DC)

Suponha uma sequência de caracteres S_1 que representa a subárvore enraizada por um nó R_1. Para ser utilizada pela função de Levenshtein, S_1 é criada através da concatenação sucessiva dos nomes dos elementos (nome da tag), apenas na abertura, durante uma navegação em profundidade na árvore. O mesmo procedimento é utilizado em um nó R_2, criando a sequência S_2. Finalmente, a função de Levenshtein é utilizada para comparar S_1 e S_2.

A sequência de caracteres do conjunto generalizador 1 : "TRTD-TRTDTDTDTDTRTDTD**AIMG**TRTDTRTD" e a do conjunto generalizador 2: "TRTDTRTDTDTDTDTRTDTDTRTDTRTD", foram geradas a partir da árvore ilustrada pela Figura 3.3, onde são apresentados dois conjuntos generalizadores de tamanho 5. Repare que, quando é necessário comparar conjuntos generalizadores com tamanho maior que 1, basta repetir o procedimento para cada raiz do conjunto generalizador e concatenar sucessivamente as sequências de caracteres geradas.

PUC-Rio - Certificação Digital Nº 0821371/CA

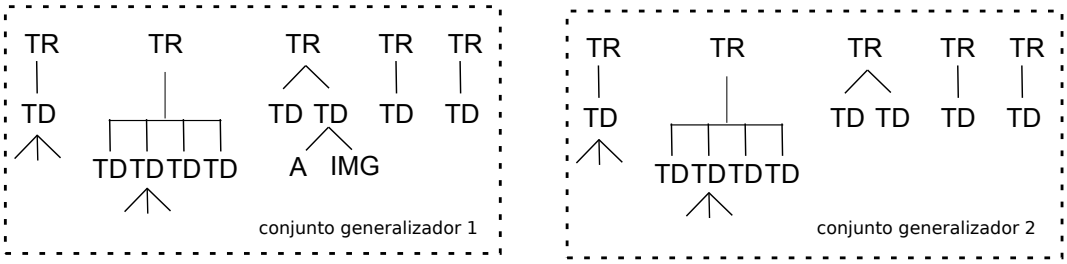

Figura 3.3: Ilustração de dois nós generalizadores de tamanho 5

A utilização desse tipo de abordagem apresenta, inicialmente, dois problemas principais. O primeiro é que durante o processamento da distância os níveis da árvore não são respeitados, então estruturas diferentes podem gerar distância igual a zero. O segundo problema é que a unidade de medida de distância na função de Levenshtein é o caractere. Com isso, nomes de tags desproporcionais podem resultar em uma grande distância entre duas árvores parecidas. Isso acontece, pois quando analisamos a estrutura de uma árvore observamos os nós. A abordagem a seguir tenta amenizar esse problema, utilizando o nome da tag como unidade de medida na função de distância.

Distância em Tags (DG)

O procedimento DG modifica a sequência de caracteres, descrita anteriormente, criando uma lista de palavras (nomes de tags). Utilizando novamente as árvores apresentadas na Figura 3.3 podemos criar as sequências de palavras [TR,TD,TR,TD,TD,TD,TD,TR,TD,TD,**A,IMG**,TR,TD,TR,TD] para o conjunto generalizador 1 e [TR,TD,TR,TD,TD,TD,TD,TR,TD,TD,TR,TD,TR,TD] para o conjunto generalizador 2. Essa nova lista de palavras passa a ser a entrada para a função de Levenshtein, no lugar de uma sequência de caracteres. Uma pequena alteração também é feita na função Levenshtein para que cada elemento da lista de palavras seja utilizado na função de comparação, no lugar do caractere.

Distância Simple Tree Matching (DT)

O terceiro e último procedimento é uma variação de um algoritmo de maior subsequência comum, proposto inicialmente em (Yang, 1991) para a comparação da árvore sintática de dois trechos de código e utilizado por (Zhai et. al., 2005) para a identificação de subestrutura semelhantes em documentos HTML. Esse procedimento é interessante pois sua formação restringe o número de mapeamentos (diminuindo a complexidade do algoritmo), respeita os níveis da árvore e utiliza o conceito de ser uma árvore ordenada. A seguir

PUC-Rio - Certificação Digital Nº 0821371/CA

são apresentadas as definições feitas por Yang e um exemplo da execução de seu algoritmo para facilitar o entendimento.

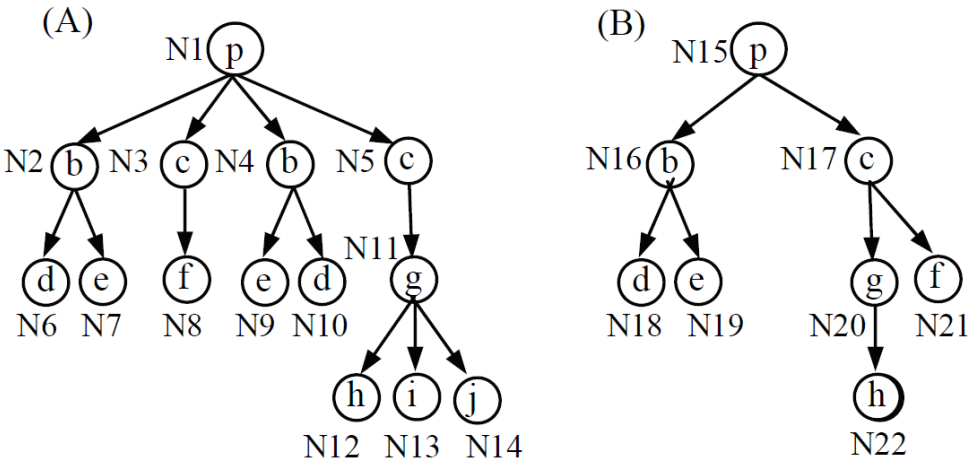

Figura 3.4: Ilustração de duas árvores A e B retirado de (Yang, 1991)

Yang define um mapeamento M entre duas árvores, A e B, como um conjunto de pares $\{(i, j)\}$, onde i e j são nós de A e B, respectivamente, e as seguintes propriedades são satisfeitas:

i Se $(i, j) \in M$ então i e j têm o mesmo símbolo (têm a mesma tag),

ii Um nó i da árvore A pertence a somente um par (i, j), ou seja é mapeado em apenas um nó da árvore B,

iii O nível de i e j são iguais e

iv Em cada nível ℓ da árvore, se i é predecessor de i' em ℓ e j é predecessor de j' em ℓ então nenhum mapeamento pode conter os pares (i, j') e (i', j) ao mesmo tempo. O mapeamento máximo é aquele que contém o maior número de pares (i, j).

A Figura 3.4 ilustra o processamento em duas árvores A e B com o procedimento da Figura 3.6. Repare que a notação $N_x - N_{x+k}$ representa uma subsequência de nós de tamanho k. O procedimento inicia comparando os nós N_1 e N_{15}. Como esses nós são iguais (têm a mesma tag), é possível criar o mapeamento $\{(N_1, N_{15})\}$ e o procedimento segue comparando o primeiro nível de filhos das árvores A e B para criar a tabela da Figura 3.5(D). O procedimento continua recursivamente comparando as subárvores enraizadas pelos nós N_2 e N_{16}. Essa etapa retorna o mapeamento máximo com cardinalidade 3, pois é possível criar os mapeamentos $\{(N_2, N_{16}), (N_6, N_{18}), (N_7, N_{19})\}$. O mapeamento máximo retornado pelo processamento das raízes N_2 e N_{17} é vazio, já que essas duas raízes contêm tags diferentes. O resultado do mapeamento entre

PUC-Rio - Certificação Digital Nº 0821371/CA

(C) $M_{1\text{-}15}$

	0	1 (N16)	2 (N16-N17)
0	0	0	0
1 (N2)	0	3	3
2 (N2-N3)	0	3	5
3 (N2-N4)	0	3	5
4 (N2-N5)	0	3	6

(D) $W_{1\text{-}15}$

	1 (N16)	2 (N17)
1 (N2)	3	0
2 (N3)	0	2
3 (N4)	2	0
4 (N5)	0	3

Figura 3.5: Ilustração da tabela final do procedimento (C) e do cálculo da primeiro nível de mapeamento (D) retirado de (Yang, 1991)

N_3 e N_{16} também é vazio. Já a cardinalidade do mapeamento entre as raízes N_3 e N_{17} é 2. Note que, quando comparamos N_4 com N_{16}, os pares (N_9, N_{19}) e (N_{10}, N_{18}) não podem ser criados simultaneamente, já que tal criação violaria a restrição (iv). O valor retornado no casamento das subárvores N_5 e N_{17} é 3, com os mapeamentos $\{(N_5, N_{17}), (N_{11}, N_{20}), (N_{12}, N_{22})\}$ finalizando o cálculo do mapeamento para o primeiro nível. Agora é possível aplicar o procedimento de programação dinâmica para calcular a tabela M (item C da figura). Ao final do procedimento temos a tabela M preenchida e o valor de $M[4, 2]$ é 6. Porém, o número total de mapeamentos criados foi 7, pois não computamos o casamento das raízes das duas árvores. O mapeamento é: $\{(N_1, N_{15}), (N_2, N_{16}), (N_6, N_{18}), (N_7, N_{19}), (N_5, N_{17}), (N_{11}, N_{20}), (N_{12}, N_{22})\}$.

Observe que o resultado do procedimento DT é o número de pares do mapeamento máximo entre duas árvores, diferente dos procedimentos anteriores. Por esse motivo, é necessário manipular o resultado obtido pelo procedimento, para ser possível obter uma unidade de distância. A abordagem proposta é de calcular a razão r entre o valor do mapeamento máximo e o número de nós da árvore que contém mais nós. A distância que utilizamos é dada por $1 - r$.

3.1.2 Funções de busca

O objetivo dos procedimentos de busca é identificar os conjuntos generalizadores para criar as *data regions*. Com isso, esses procedimentos de busca organizam as comparações, ou seja, a forma com que os nós são agrupados sem que o custo computacional para isso seja muito alto.

PUC-Rio - Certificação Digital Nº 0821371/CA

```
Procedimento DT(A, B):
1.  Se o nome do elemento do nó A e B são diferentes, então:
2.      Retorne 0

3.  Sejam A' e B' as listas de filhos dos nós A e B respectivamente e
4.  M[|A'|][|B'|] uma matriz com |A'| colunas e |B'| linhas

5.  Para i variando de 1 até |A'| + 1:
6.      Para j variando de 1 até |B'| + 1:
7.          w ← DT(A'[i], B'[j])
8.          M[i][j] ← max(M[i][j − 1], M[i − 1][j], M[i − 1][j − 1] + w)

9.  Retornar M[|A'|][|B'|] + 1
```

Figura 3.6: Procedimento Simple Tree Matching

Casamento Simples (CS)

O procedimento de busca Casamento Simples (CS), apresentado na Figura 3.7, tenta criar *data regions*, assumindo que cada nó é um conjunto generalizador, ou seja, todos os conjuntos generalizadores têm tamanho igual a 1. Esse procedimento recebe como entrada uma lista L de nós irmãos na árvore DOM e particiona essa lista em sub-listas de nós $C_1, .., C_p$ de modo que as subárvores enraizadas em nós consecutivos de uma sub-lista C_i tenham estruturas semelhantes. Para medir essa semelhança podemos utilizar qualquer função de distância apresentada na Seção 3.1.1. No final do processamento, as sub-listas que contêm mais de um nó são classificadas como *data regions*.

A abordagem do procedimento CS é simples porém apresenta resultados interessantes nas tarefas relatadas nos Capítulos 4 e 5.

```
Procedimento CS(L: lista de nós):
1.  k ← 0 e C_k ← L_1
2.  Para i variando de 2 até |L|:
3.      Se a distância entre L_{i−1} e L_i for menor ou igual a α, então:
4.          C_k ← C_k ∪ L_i
5.      Senão:
6.          k ← k + 1 e C_k ← L_i
```

Figura 3.7: Procedimento Casamento Simples

O procedimento casamento simples pode ser diretamente comparado com o procedimento descrito em (Liu et. al., 2003). O algoritmo de Liu *et. al.* compara inicialmente todos os nós consecutivos, um a um, utilizando a função de Levenshtein para calcular a distância entre as subárvores. Em seguida os nós são agrupados dois a dois sem sobreposição, criando os conjuntos

PUC-Rio - Certificação Digital Nº 0821371/CA

generalizadores de tamanho 2, para que o processo de cálculo da distância possa ser repetido, e assim sucessivamente. Os grupos de conjuntos generalizadores guardados, que determinam as data regions, são aqueles que agregam mais nós, ou seja, têm os conjuntos generalizadores de maior tamanho.

Após avaliar o algoritmo apresentado em (Liu et. al., 2003), podemos dizer que o procedimento CS é uma simplificação do algoritmo proposto, onde os nós são comparados sequencialmente um a um e a maximização não é realizada. Tal adaptação diminui o custo computacional do algoritmo e apresenta resultados interessantes, como será mostrado nos Capítulos 4 e 5. O procedimento CS é proposto como uma solução rápida para a busca de *data region*.

Casamento de Árvores

O procedimento Casamento de Árvores (CA) é um pouco mais elaborado que o CS e busca identificar as *data regions* que são formadas por estruturas mais complexas, ou seja, estruturas que são formadas por conjuntos generalizadores com tamanho maior que 1. Isso acontece, pois diversas estruturas repetidas podem apresentar elementos de separação que na árvore DOM são adicionados como nós intermediários às estruturas, como na Figura 3.8. Repare que para cada item descrito em uma *tag table* existe um link, *tag a*, criando um intervalo entre os nós que descrevem cada item. Nesse caso, é necessário criar conjuntos generalizadores de tamanho 2, para ser possível identificar as *data regions*.

O procedimento CA recebe como entrada uma lista de nós $L_1, ..., L_n$ situados no mesmo nível da árvore DOM. Inicialmente o procedimento busca o menor $p \geq 2$ tal que a distância entre L_1 e L_p seja menor que α. A motivação é identificar estruturas com nós generalizados maiores que 1, já que o procedimento CS (descrito na Seção 3.1.2) é especializado em *data regions* formados por nós generalizadores de tamanho 1. Se tal p não existe, o procedimento busca *data regions* recursivamente na lista de nós $L_2, \ldots, L_n$. Caso contrário, o procedimento define $G_1 = \{L_1 \ldots, L_{p-1}\}$ como um conjunto generalizador e busca outros conjuntos generalizadores semelhantes a G_1. Isso é feito buscando o menor k tal que a distância entre L_{kp} e $L_{(k+1)p}$ seja maior que α. A partir desse k, o procedimento define os conjuntos generalizadores $G_1, \ldots, G_k$, onde $G_i = \{L_{1+(i-1)p}, \ldots, L_{ip-1}\}$, e verifica se esses formam uma *data region*. Essa verificação consiste em utilizar uma das funções de distância descritas anteriormente, para validar a Condição 4 da Definição 2 (a distância entre dois conjuntos generalizadores deve ser menor que α). Finalmente, o procedimento busca data regions recursivamente na lista de nós

PUC-Rio - Certificação Digital Nº 0821371/CA

$L_{(k+1)p}, \ldots, \ldots, L_n$.

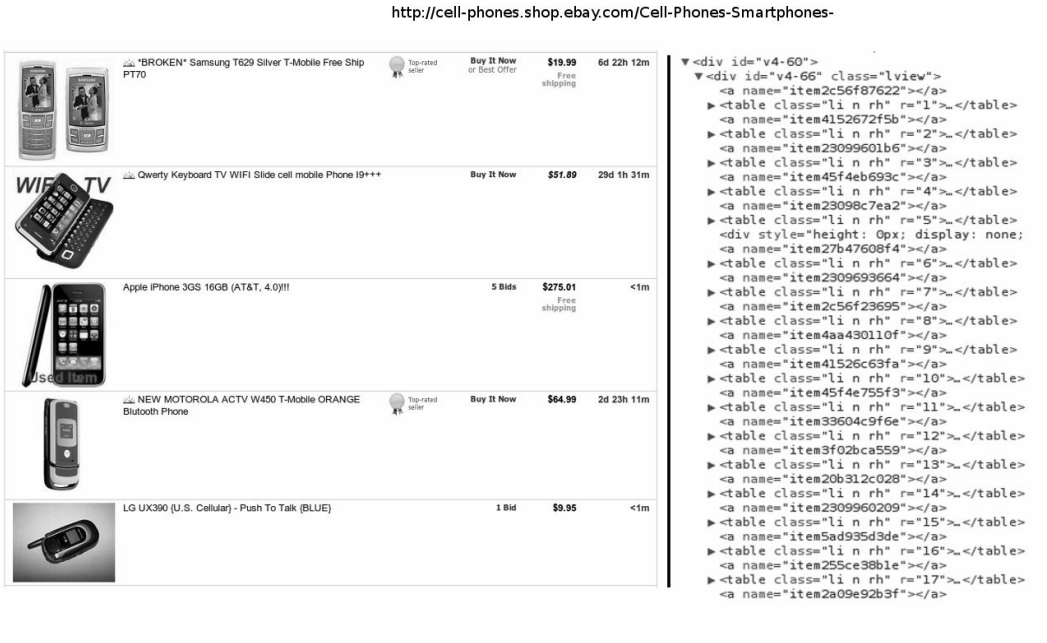

Figura 3.8: Exemplo de estrutura que utiliza um conjunto generalizador de tamanho 2 para apresentar os itens

3.2 O ambiente de experimentação

Existem diversos problemas não solucionados que envolvem a recuperação, segmentação e identificação de informação de páginas HTML. Por serem referentes às páginas disponíveis na Web, normalmente, esses problemas demandam a experimentação em conjuntos de controle, para que se possa avaliar o comportamento da solução proposta antes de colocá-la em produção.

A experimentação em documentos HTML pode ser uma tarefa trabalhosa pois, para atacar o problema proposto, é necessário enfrentar diversas dificuldades para que o documento HTML possa ser carregado em memória e analisado. Por exemplo, diferentes codificações de caracteres ou má geração do HTML são esperados dentre os documentos da Web. Esses problemas podem exigir profundo conhecimento de aspectos como a construção da linguagem ou tabelas de codificação que, em grande maioria, fogem do domínio da tarefa principal desejada.

Após sanar todos os problemas referentes ao tratamento dos documentos HTML, surgem novos desafios, que mesmo tendo um vínculo maior com a tarefa desejada, poderiam ser evitados. Geralmente, a maneira de armazenar o conjunto resposta, assim como avaliar o resultado que foi obtido, são problemas

PUC-Rio - Certificação Digital Nº 0821371/CA

enfrentados por todos que desejam realizar experimentos com documentos HTML. Porém, na maioria dos trabalhos apresentados a abordagem utilizada é semelhante, o que sugere espaço para a criação de um conjunto de ferramentas públicas que facilite a tarefa de experimentação e avaliação dos resultados.

O cenário descrito nos parágrafos anteriores é encontrado ao serem estudados os trabalhos relacionados à recuperação de informação e segmentação em documentos HTML. Com isso, é proposta uma ferramenta para facilitar futuros trabalhos que necessitem realizar experimentação em páginas HTML.

As seções seguintes apresentarão os modelos e características necessárias para uma ferramenta de suporte à experimentação em documentos HTML.

3.2.1 Modularização da ferramenta

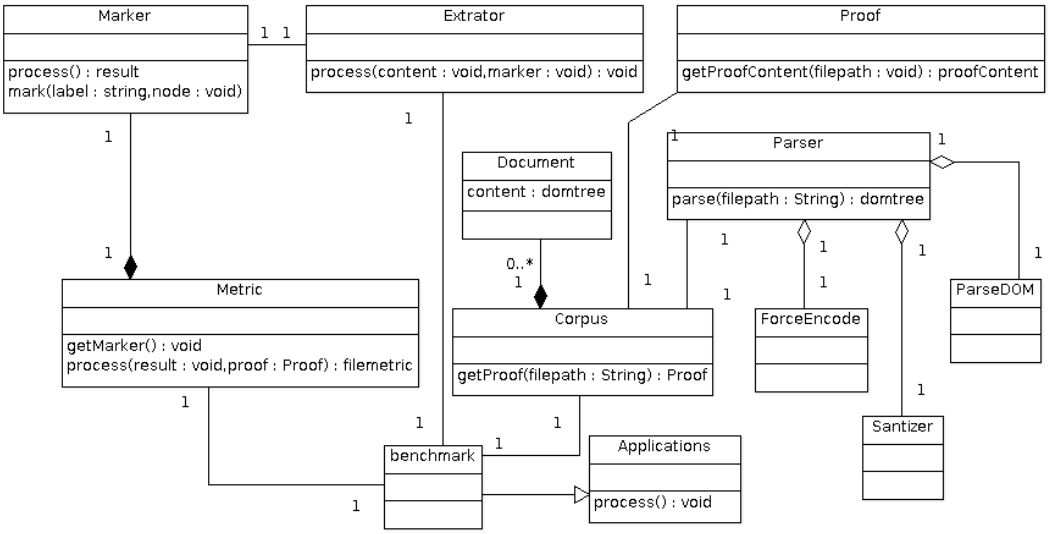

Figura 3.9: Diagrama de classes da ferramenta

Observando os passos necessários para que seja possível o desenvolvimento de uma ferramenta de experimentação, fica evidente a necessidade de um módulo capaz de corrigir problemas de codificação e má formação nos documentos HTML. Além disso, é necessária também a existência, em memória, de uma estrutura representativa do documento HTML tornando possível a manipulação desse por diversos algoritmos.

O módulo Parser atende essas necessidades, sendo dividido em três submódulos. O ForceEncode normaliza a codificação de um documento HTML para que esse contenha todos os seus caracteres em uma codificação padrão. O Sanitize verifica a corretude do documento HTML e corrige alguns erros e problemas comumente encontrados nos documento HTML, como não fechamento

PUC-Rio - Certificação Digital Nº 0821371/CA

de algumas tags, tags que não são definidas em algumas versões da HTML que ocasionam problemas durante a criação da estrutura em memória do documento. O ParserDOM é o último componente do módulo Parser, e pode ser considerado o mais importante, sendo o procedimento que transforma o documento HTML em uma estrutura em memória DOM

Com a estrutura do documento em memória, é possível executar rotinas/algoritmos sobre tal estrutura. A maneira como as rotinas devem armazenar os resultados é importante, pois determina todo o processo de geração de resultados e avaliação. Com isso, o módulo Marcadores é responsável por armazenar as informações importantes, sendo responsável por gerir as informações necessárias para identificar rótulos de nós.

O Módulo Extratores é um usuário direto dos Marcadores, pois os extratores são os procedimentos/heurísticas que adicionam marcações, ou seja, associam rótulos aos nós de uma árvore DOM. Os extratores respeitam uma interface para que esses possam ser facilmente conectados ao framework, podendo utilizar todo o suporte para o processamento do documento HTML.

Existe uma peculiaridade com a modelagem adotada, pois a forma de marcar o resultado influencia na sua avaliação, porém essa dependência será descrita junto com o módulo de avaliação.

Os módulos descritos fornecem um conjunto de funcionalidades que proporcionam a resolução de uma tarefa de identificação. Porém, como o objetivo da ferramenta é proporcionar um ambiente de experimentação, são necessários módulos para realizar a tarefa de experimentação.

O módulo Avaliador fornece o conjunto de regras para que o resultado obtido possa ser mensurado e retornado como informação estatística. As formas de avaliar são diretamente relacionadas à forma que o resultado foi armazenado. Assim, para cada avaliador existe um marcador disponível. Esse marcador armazena a solução no formato adequado para ser posteriormente avaliado.

Além do Marcador, o avaliador necessita também de uma fonte de informação correta que é o Gabarito. O Gabarito fornece um conjunto de informações, no mesmo formato que o conjunto armazenado pelo marcador, sendo que esse contém a marcação correta para que os Marcadores possam ser avaliados. Um formato básico que atende às principais tarefas de identificação em páginas HTML é fornecido junto à ferramenta.

A experimentação normalmente ocorre sobre um grande conjunto de documentos. Por esse motivo, foi criado o Corpus, uma estrutura que descreve esse conjunto. O Corpus descreve um conjunto de documentos, informando sua natureza, o tipo de gabarito que pode ser obtido dos documentos e para quais

PUC-Rio - Certificação Digital Nº 0821371/CA

tarefas esses documentos são interessantes.

Com esse conjunto de módulos disponível, diversas aplicações, como *benchmark*, podem ser implementadas para utilizar tais módulos. *Benchmark* é uma aplicação que gera um relatório de desempenho para um conjunto de extratores e corpus, utilizando os módulos mencionados anteriormente. O *benchmark* é a única aplicação integrada à ferramenta, pois é uma necessidade direta para a análise da experimentação. Na Figura 3.10, é apresentado um diagrama de sequência para exemplificar a utilização da ferramenta.

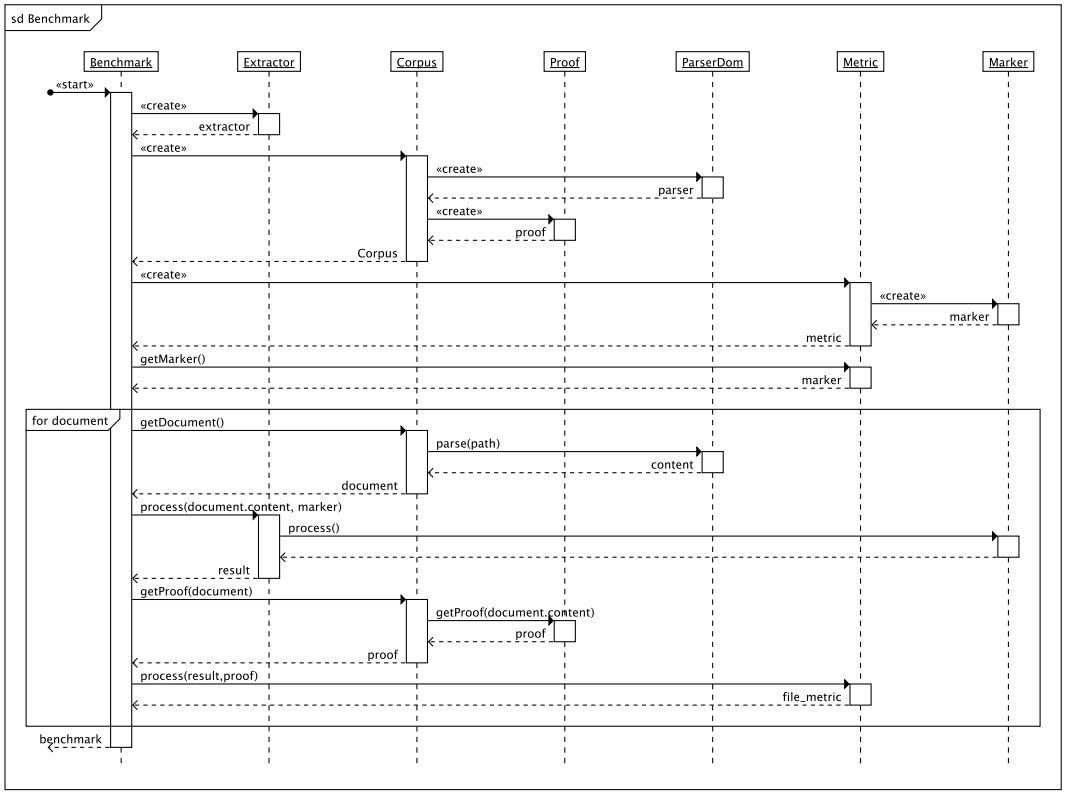

Figura 3.10: Diagrama de sequência da aplicação Benchmark

PUC-Rio - Certificação Digital Nº 0821371/CA