Representação em XML da Floresta Sintáctica

Rui Vilela[1], Alberto Simões[1,2], Eckhard Bick[3], and José João Almeida[2]

[1]Linguateca, pólo de Braga
{ruivilela|ambs}@di.uminho.pt

[2]Departamento de Informática, Universidade do Minho
{ambs|jj}@di.uminho.pt

[3]Institute of Language and Communication, University of Southern Denmark
eckhard.bick@mail.dk

Resumo A Floresta Sintáctica é um recurso linguístico mantido e distribuído livremente ao público pela Linguateca. Face à necessidade de abranger uma maior comunidade de linguistas e de programadores na área do processamento da linguagem natural, procura-se distribuir este recurso em outros formatos alem do existente, tais como formatos baseados em XML.
Pretende-se neste documento descrever o processo de equipar a Floresta Sintáctica com mecanismos que facilitem a sua utilização. Para isso pretende-se converter o seu formato actual para formatos baseados em XML. Serão analisados os problemas e questões da definição XML deste tipo de recurso.
Apresenta-se ainda um módulo experimental construído especificamente para facilitar a construção de processadores da Floresta.

1 Introdução

Pretende-se representar em XML uma estrutura organizada de textos anotados sintacticamente em forma de árvore, referenciada como *treebank*. De uma forma simplicista, um *treebank* pode ser comparado a um conjunto de árvores sintácticas de diferentes frases.

A necessidade de um recurso semelhante a um*treebank* na área do processamento computacional da língua portuguesa, que contenha uma base de dados estruturada de textos, previamente analisados e classificados gramaticalmente, levou ao surgimento da Floresta Sintáctica [1], um recurso disponibilizado pela Linguateca [13].

1.1 A Linguateca

A Linguateca[1] foi iniciada por iniciativa de Diana Santos [10], trata-se de um Centro de Recursos distribuído para a Língua Portuguesa. O objectivo da Linguateca é promover a existência, discussão, avaliação, distribuição e manutenção

[1] http://www.linguateca.pt

de recursos relacionados com o processamento linguístico do Português a uma comunidade que os utiliza. A Linguateca é parcialmente financiada pela Fundação para a Ciência e Tecnologia de Portugal através do registo POSI/PLP/43931/2001, e co-financiada pelo projecto POSI.

1.2 A Floresta Sintáctica

A Floresta Sintáctica nasceu de um projecto de colaboração entre a Linguateca, pólo de Oslo, e o projecto VISL[2], cujos responsáveis são Diana Santos e Eckhard Bick respectivamente. Actualmente é constituída por texto jornalístico português e brasileiro, cedidos pelo Jornal Público e Folha de São Paulo.

A Floresta Sintáctica está publicamente disponível na Linguateca[3]. Sendo um corpus analisado sintacticamente (um conjunto de árvores sintácticas), a Floresta (ou um *treebank* em geral) constitui um relevante contributo para a comunidade já que [11]:

- foi revista e corrigida manualmente;
- está bem documentada, mantida, e com ferramentas associadas;
- reflecte um consenso entre a comunidade linguística;
- constitui um recurso para avaliar ou para avaliação, já que é facilmente transformável em algo que sirva de padrão para comparação;
- pode ser usada em processos de aprendizagem automática ou assistida de ferramentas de *parsing* ou *tagging*;
- permite a realização de estudos linguísticos, nomeadamente no campo sintáctico;

A Floresta Sintáctica é desenvolvida de forma a ser um recurso computacional para aplicações mais complexas, contendo a maior quantidade possível de informação sintáctica útil [12].

As árvores que constituem a Floresta Sintáctica são geradas automaticamente pelo programa PALAVRAS [4]. Posteriormente, cada árvore é revista por linguistas. A revisão manual de cada frase é exaustiva e demorada, mas garante uma qualidade superior na classificação sintáctica das árvores.

O formato base usado na Floresta Sintáctica é chamado árvores deitadas (ad). A nomenclatura deste formato é descrita ao pormenor em [2]. Segue-se um pequeno exemplo.

```
1 SOURCE: CETENFolha n=22 cad="Cotidiano" sec="soc" sem="94a"
2 CF22-3 Escuto Stones desde os 13 anos de idade.
3 A1
4 STA:fcl
5 =P:v-fin('escutar' PR 1S IND)   Escuto
6 =ACC:prop('Stones' M P) Stones
7 =ADVL:pp
8 ==H:prp('desde')        desde
```

[2] http://visl.sdu.dk/visl/pt

[3] http://www.linguateca.pt/Floresta

```
==P<:np
===>N:art('o' <artd> M P)       os
===>N:num('13' <card> M P)      13
===H:n('ano' M P)       anos
===N<:pp
====H:prp('de') de
====P<:n('idade' F S)   idade
=.
```

Seguindo o exemplo, a primeira linha contém informação relativa à secção de texto de onde foi retirada a frase, incluindo o contexto.

A segunda linha contém o identificador único da frase, com a respectiva frase (CF para CETENFolha da Folha de São Paulo, CP para CETEMPúblico do Público).

`A1` define uma secção onde foi elaborada uma análise sintáctica. É colocada antes do nó raiz da árvore. Uma árvore pode conter várias análises sintácticas distintas sendo estas numeradas consequentemente por A2, A3, A4. Cada secção analisada sintacticamente é separada por &&.

O nó raiz, que está definido na 4ª linha, inicia a construção da árvore. Para identificar os níveis da árvores é usado o símbolo '='. Cada símbolo adicional representa mais um nível de profundidade na árvore (esquematizada na figura 1. Cada nó que tenha um incremento de nível na linha seguinte é um nó não terminal.

Todos os nós contêm informação morfossintáctica, constituída por duas etiquetas separadas por ':'. As etiquetas do lado esquerdo (STA,ADVL,H,N<) são etiquetas de *função*, enquanto que as do lado direito (fcl, pp, prp) são etiquetas de *forma*. Os nós podem conter informação morfológica e gramatical associada às palavras e/ou à frase em que se inserem, como por exemplo, informação morfológica, lema da palavra, e etiquetas secundárias.

1.3 Processamento da Floresta Sintáctica

O formato árvores deitadas não é fácil de ser processado computacionalmente. Actualmente, a floresta é composta por aproximadamente 7500 árvores. As dificuldades subjacentes são:

- uma quantidade extensa e complexa de etiquetas de informação morfossintáctica que deve ser devidamente validade;
- o facto de ser produto de uma revisão manual, está sujeito a conter erros na sua sintaxe;
- a actualização periódica da Floresta Sintáctica obriga a re-processamentos.

O formato base da floresta, não permite facilmente uma extracção de informação eficiente. Para facilitar o acesso a este recurso, é necessário exportar para outros formatos a partir do formato base. Estes novos formatos devem suportar as características do original, oferecendo uma sintaxe mais simples e/ou uma API de programação.

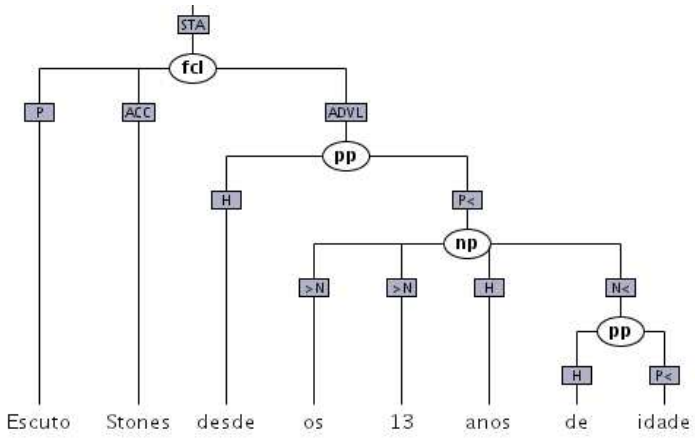

Figura 1. Representação gráfica de uma árvore.

2 Exportação da Floresta Sintáctica

Já existem diversos formatos, não baseados no XML, semelhantes ao formato base da floresta, desenvolvidos noutros países por projectos internacionais semelhantes, tais como: SUSANNE [9], Peen Treebank [8] e Tiger [6].

Estes formatos foram inicialmente desenvolvidos tendo em conta a morfologia gramatical das línguas a que se aplicavam, e o nível de detalhe exigido pelos linguistas e programadores para classificação sintáctica. Tendo em conta o último aspecto, não é fácil a exportação para outros formatos devido principalmente a:

- eliminação de informação morfológica originalmente contida pelo formato original. Resulta na perda de informação considerada importante em termos linguísticos;
- alteração da notação para indexar a floresta. Se possível, é desejável manter à parte a notação original;
- o novo formato contêm regras definidas rigidamente pelos autores;
- dificuldade em adaptar a informação morfológica original. Alguns dos formatos existentes não estão preparados para certos tipos de classificação morfológica;
- ferramentas inadequadas ou insuficientes para explorar o formato.

Qualquer um destes pontos inviabiliza a exportação correcta da Floresta Sintáctica nos formatos mencionados.

Dada a facilidade de estruturação do XML, pretende-se definir um formato que consiga reter toda a informação da floresta sintáctica.

3 A Floresta Sintáctica em XML

Nesta secção apresentamos duas abordagens à representação da floresta sintáctica em XML.

3.1 Tiger-XML

O formato Tiger-XML foi desenhado como uma linguagem de interface para o formato Tiger [6]. Este formato possui uma ferramenta para pesquisa, designada por TIGERSearch [7].

O TIGERSearch permite explorar sintacticamente corpora anotado, mediante a visualização do *treebank*. Pode-se obter fenómenos relevantes sintacticamente a partir de frases, propriedades lexicais e contexto da palavra na frase. Actualmente pode servir de interface a alguns *treebanks* existentes [7].

Para converter o formato base da Floresta Sintáctica para o formato Tiger-XML é usado um programa em Perl, originalmente desenvolvido pelo Eckhard Bick. Este programa foi modificado sucessivamente de forma a adaptar-se à Floresta Sintáctica.

Segue-se a descrição do processo de conversão:

- conversão do formato árvores deitadas (ad) para Tiger-XML em *bruto*;
- análise do código XML gerado, eliminando elimina secções inválidas de XML. Estas secções inválidas ocorrem geralmente devido a erros existentes no formato base;
- geração automática da *header* do XML, contendo a nomenclatura das etiquetas de função e forma das palavras. O *script* identifica e recolhe os valores dos atributos existentes no código XML. Todos os valores destes atributos estão descritos num ficheiro de texto. Se os valores dos atributos já possuírem uma descrição definida, são adicionados ao header do XML. Este processo ajuda a identificar árvores que possuam atributos errados. Este processo também ajuda a identificar árvores com erros no formato base;
- O processo é finalizado pela indentação do código XML usando a ferramenta de domínio público *xmllint*.

Durante o processo é adquirida informação estatística e identificados erros, o que permite o desenvolvimento de ferramentas de validação sobre o formato das árvores deitadas. Segue-se um exemplo de uma árvore no formato Tiger-XML:

```
<s id="s85" ref="CF22-3"
      source="CETENFolha n=22 cad=Cotidiano sec=soc sem=94a"
      forest="1" text="Escuto Stones desde os 13 anos de idade.">
  <graph root="s85_500">
    <terminals>
      <t id="s85_2" word="Escuto" lemma="escutar" pos="v-fin"
        morph="PR 1S IND" extra="--" />
      <t id="s85_3" word="Stones" lemma="Stones" pos="prop"
        morph="M P" extra="--" />
```

```
          <t id="s85_4" word="desde" lemma="desde" pos="prp"
            morph="--" extra="--" />
          <t id="s85_5" word="os" lemma="o" pos="art"
            morph="M P" extra="artd" />
          <t id="s85_6" word="13" lemma="13" pos="num"
            morph="M P" extra="card" />
          <t id="s85_7" word="anos" lemma="ano" pos="n"
            morph="M P" extra="--" />
          <t id="s85_8" word="de" lemma="de" pos="prp"
            morph="--" extra="--" />
          <t id="s85_9" word="idade" lemma="idade" pos="n"
            morph="F S" extra="--" />
          <t id="s85_10" word="." lemma="--" pos="pu"
            morph="--" extra="--" />
        </terminals>
        <nonterminals>
          <nt id="s85_500" cat="s">
            <edge label="STA" idref="s85_501" />
          </nt>
          <nt id="s85_501" cat="fcl">
            <edge label="P" idref="s85_2" />
            <edge label="ACC" idref="s85_3" />
            <edge label="ADVL" idref="s85_502" />
          </nt>
          <nt id="s85_502" cat="pp">
            <edge label="H" idref="s85_4" />
            <edge label="P<" idref="s85_503" />
          </nt>
          <nt id="s85_503" cat="np">
            <edge label=">N" idref="s85_5" />
            <edge label=">N" idref="s85_6" />
            <edge label="H" idref="s85_7" />
            <edge label="N<" idref="s85_504" />
          </nt>
          <nt id="s85_504" cat="pp">
            <edge label="H" idref="s85_8" />
            <edge label="P<" idref="s85_9" />
          </nt>
        </nonterminals>
      </graph>
    </s>
```

Este formato é particularmente difícil de utilizar programaticamente por conter secções diferentes relativas aos símbolos terminais e não terminais (o que obriga à criação de tabelas de indirecção).

3.2 SimTreeML

O formato Tiger-XML tem várias vantagens mas as suas árvore não estão directamente identificáveis.

O formato SimTreeML (exemplo seguinte) contém uma representação dos níveis da árvores mais fácil de identificar, e tem uma notação mais compacta. O extracto seguinte descreve em SimTreeML a mesma árvore da secção anterior.

```
<extract>
  <source>CF22-3 Escuto Stones desde os 13 anos de idade.</source>
  <tree cat='fcl' fun='STA'>
     <t cat='v-fin' lema='escutar' args='PR 1S IND' fun='P'>Escuto</t>
     <t cat='prop' lema='Stones' args='M P' fun='ACC'>Stones</t>
     <tree cat='pp' fun='ADVL'>
        <t cat='prp' lema='desde' fun='H'>desde</t>
        <tree cat='np' fun='P<'>
           <t cat='art' lema='o' args='M P' fun='>N' extra='artd'>os</t>
           <t cat='num' lema='13' args='M P' fun='>N' extra='card'>13</t>
           <t cat='n' lema='ano' args='M P' fun='H'>anos</t>
           <tree cat='pp' fun='N<'>
              <t cat='prp' lema='de' fun='H'>de</t>
              <t cat='n' lema='idade' args='F S' fun='P<'>idade</t>
           </tree>
        </tree>
     </tree>
     <punt ort='.'/>
  </tree>
</extract>
```

Este tipo de representação XML, permite um fácil processamento no sentido de construir ou visualizar as árvores.

Uma das funcionalidades óbvias exigida a qualquer processador de extractos da floresta é o permitir a visualização de árvores através da definição de um conjunto de CSS ou XSLT (correspondentes a um conjunto de vistas eficazes).

Na figura 2 mostra-se a visualização directa de árvores da Floresta no formato descrito, usando apenas um *browser* (Mozilla) e uma CSS. Esta CSS desenha uma área em volta de cada **tree** e esconde alguns dos atributos da informação (para não sobrecarregar o utilizador com demasiada informação).

3.3 Comparação dos formatos TigerXML – SimTreeML

Dum modo simplificado a abordagem TigerXML corresponde a codificar a informação da Floresta como sendo um grafo (descrevendo-se os seus nós e ramos). A abordagem SimTreeML corresponde a descrever árvores *puras*.

Os grafos TigerXML têm uma leitura mais complexa em XML do que as árvores SimTreeML, no entanto oferecem um maior poder expressivo (que pode ser usado para enriquecer futuramente as árvores anotadas com informação adicional semântico-linguística diversa como por exemplo ligação entre os pronomes e os nomes que referem).

O TigerXML tem a vantagem de ser um formato bastante usado e difundido e dispor de um conjunto de ferramentas de suporte como sejam o TigerSearch[7].

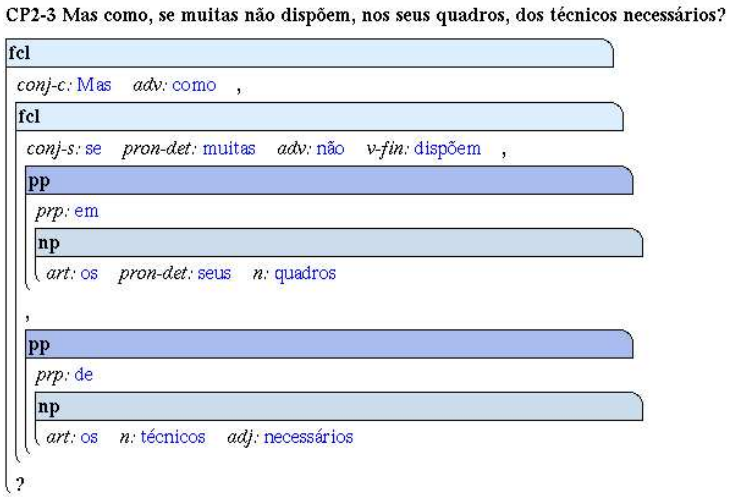

Figura 2. Extracto formatado com uma CSS de uma árvore da floresta no formato SimTreeML

Por outro lado, juntamente com os tradutores para o formato SimTreeML foi construído um módulo Perl (`Lingua::TreeBank::Dirty`), que facilita a construção de processadores da Floresta Sintáctica a partir deste formato.

4 O módulo Lingua::TreeBank::Dirty

O módulo Perl `Lingua::TreeBank::Dirty` usa a Floresta em formato SimTreeML para construir um conjunto de estruturas internas que tornem viável a criação de processadores usando métodos habituais em XML, mas que não obriguem a ter a totalidade da estrutura em memória principal.

Um dos objectivos deste módulo é que a descrição dos processadores seja feita dum modo tão compacto e simples quanto possível.

Nesta secção apresentaremos apenas dois exemplos baseados na função `downTr` (*down translate*), com o intuito de mostrar que é possível construir com pouco esforço pequenos programas específicos que dificilmente encontraríamos em aplicações de tratamento de corpora anotados.

4.1 Exemplo para estudos *cognitivos*

Por vezes em estudos cognitivos é por vezes necessário procurar palavras relacionadas com outras em certos contextos de uso.

Neste exemplo (um pouco artificial) construiu-se um programa Perl usando o `Lingua::TreeBank::Dirty` que, dado um verbo (comer), procura todas os extractos que contêm esse lema e processa a respectiva frase extraindo os nominais nela inscluídos.

Este processador imprime todas as palavras cuja categoria (`$v{cat}`) seja `n` (i.e. os nominais)

O `Lingua::TreeBank::Dirty` usa internamente o módulo XML::DT [3,14], herdando dele uma variedade de funções (que aqui não serão detalhadas) que permitem consulta do contexto, alteração de subárvores e de atributos.

Dum modo simples, a construção dum processador é feito através de:

- difinição de um padrão textual de domínio de interesse;
- definição de uma função de processamento para os símbolos terminais (palavras);
- definição de uma função de processamento para os nós internos da árvore sintáctica.

Cada uma destas funções recebe como parámetro os atributos (%v) e o conteúdo ($c).

```
1  use Lingua::TreeBank::Dirty "/home/jj/linguateca/FS";

2  downTr({ -patt => "-w comer",
3            -end => sub{print "\n"},
4               t => sub{print "$c, "  if $v{cat} eq "n" } });
```

Notas:

- linha 2 – selecciona os extractos contendo o lema “comer”. Naturalmente que a aplicação deste padrão tem consequências na quantidade de árvores a analisar e portanto no tempo total de execução.
- linha 3 – mudar de linha após processar cada extracto;
- linha 4 – imprimir as palavras que estejam classificadas na categoria “n”.

Quando aplicado à Floresta Sintáctica, a saída do programa anterior é:

```
1  carvão, restaurante, tribunal, carne,
2  invernos, caminhos, aldeia, tangerinas, cascas, montes, neve,...
3  praia, colina, copos, água, restaurante, trutas, castanhas, lado,
4  milhões, pessoas, música, relações, mês, muçulmano,
5  ......
6  arroz, marisco,
7  vinho, vinho, milhão, portugueses, crianças, sopas, cavalo,
8  bicicleta, sacos-cama, roupa, dinheiro,
9  pinguim, carne, bode, magote, sertanejos,
10 tipos, refeições, símbolos, valor, pessoas,
```

Como se constacta, foi possível em poucas linhas fazer a construção de um processador. Esta facilidade viabiliza a construção de processadores para resolver situações pontuais e esporádicas, i.e., ajuda à existência de uma bancada de trabalho onde se possa tirar partido da Floresta.

4.2 Exemplo gramatical

Neste exemplo processa-se (uma parte de) a Floresta para fazer a extracção de uma gramática implícita.

Cada ramificação corresponde ao uso de uma produção. Faz-se também a acumulação e contagem das produções encontradas.

```
downTr( { tree => sub{ $p = "$v{cat} -> $c";
                       $prod{$p}++;
                       $v{cat} },
             t => sub{ "[$v{cat}]" },
          punt => sub{ $v{ort} }
        });

#... imprimir ordenadas por ordem inversa de ocorrências
```

Este programa quando aplicado à Floresta, constrói um ficheiro de produções sendo as primeiras linhas as seguintes:

```
pp ->  [prp] np           17250
np ->  [art] [n]          5670
np ->  [art] [n] pp       4621
np ->  [art] [prop]       2843
pp ->  [prp] [n]          2224
np ->  [n] pp             1768
pp ->  [prp] icl          1398
pp ->  [prp] [prop]       1302
np ->  [art] [n] [adj]    1221
np ->  [pron-det] [n]     1107
vp ->  [v-fin] [v-pcp]    749
np ->  [n] [adj]          718
...
fcl ->  np [v-fin] np .   188
```

5 Conclusões

A conversão entre formatos permitiu a descoberta localizada de diversos erros na sintaxe do formato árvores deitadas. Assim, desenvolveram-se *scripts* para validação da Floresta. Os relatórios de erros encontrados podem ser enviados aos linguistas que revêm a Floresta.

A distribuição da Floresta Sintáctica em diversos formatos permite abranger um maior número de pessoas que estudam e desenvolvem trabalho no processamento do português, assim como um maior leque de possibilidades para o acesso e manipulação deste recurso.

Para além das ferramentas clássicas de consulta, visualização e estatísticas, salienta-se a importância da construção de módulos que facilitem a escrita de processadores novos. Na verdade, é nossa convicção que é importante associar

a cada formato em que se codifique a floresta, um conjunto de ferramentas que tornem simples e eficiente o seu processamento.

Além da comparação do Tiger-XML com o SimTreeML, seria ainda importante fazer a comparação com um terceiro formato, usado pelo XML Corpus Encoding Standard[5].

Referências

1. Susana Afonso. A floresta sintá(c)tica como recurso. Documentação disponível na Linguateca, 2003.
2. Susana Afonso. *Árvores deitadas: Descrição do formato e descrição das opções de análise na Floresta Sintá(c)tica*, 2004.
3. J.J. Almeida and José Carlos Ramalho. XML::DT a perl down-translation module. In *XML-Europe'99, Granada - Espanha*, Maio 1999.
4. Eckhard Bick. The parsing system palavras, automatic grammatical analysis of portuguese in a constraint grammar framework. 2000.
5. Vassar College Department of Computer Science. XCES — Corpus Encoding Standard for XML. `http://www.xml-ces.org/`.
6. Stefanie Dipper, Thorsten Brants, Wolfgang Lezius, Oliver Plaehn, and George Smith. The tiger treebank. In *Third Workshop on Linguistically Interpreted Corpora*, 2001.
7. Esther König, Wolfgang Lezius, and Holger Voormann. *TIGERSearch 2.1 User's Manual*, 2003.
8. Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewic. Building a large annotated corpus of english: The penn treebank. In *Computational Linguistic*, volume 19, pages 313–330, June 2003.
9. Geoffrey Sampson. *SUSANNE Corpus and Analytic Scheme*, June 2004.
10. Diana Santos. O projecto processamento computacional do português: Balanço e perspectivas. In Maria das Graças Volpe Nunes, editor, *V Encontro para o processamento computacional da língua portuguesa escrita e falada*, pages 105–113, São Paulo, 2000. ICMC/USP.
11. Diana Santos. The floresta experience. Presentation for the Swedish Treebank Symposium Växjö, Novembro 2002.
12. Diana Santos. Timber! issues in treebank building and use. In Nuno J. Mamede, Jorge Baptista, Isabel Trancoso, and Maria das Graças Volpe Nunes, editors, *Computational Processing of the Portuguese Language, 6th International Workshop*, pages 151–158. Springer Verlag, 2003.
13. Diana Santos, Alberto Simões, Ana Frankenberg-Garcia, Ana Pinto, Anabela Barreiro, Belinda Maia, Cristina Mota, Débora Oliveira, Eckhard Bick, Elisabete Ranchhod, José João Dias de Almeida, Luís Cabral, Luís Costa, Luís Sarmento, Marcirio Chaves, Nuno Cardoso, Paulo Rocha, Rachel Aires, Rosário Silva, Rui Vilela, and Susana Afonso. Linguateca: um centro de recursos distribuído para o processamento computacional da língua portuguesa. In IBERAMIA 2004, editor, *Workshop on Linguistic Tools and Resources for Spanish and Portuguese*, 2004.
14. Alberto Simões. XML::DT - down translating xml. *The Perl Review*, 1(1), 2004.