Técnicas em software livre para exploração de corpora do português livremente disponíveis na WWW

Leonel Figueiredo de Alencar – Universidade Federal do Ceará (UFC)

RESUMO: Este artigo aborda a lingüística de corpus como subárea da informática aplicada que tem na extração automática de dados de corpora um de seus focos principais. Com esse propósito, desenvolvemos comandos e scripts na linguagem de comandos `bash` do UNIX, exemplificando a sua aplicabilidade na investigação do sufixo *-vel* e de repetições de letras e palavras em dois dos principais corpora do português. Argumentamos que a utilização de software livre com interface textual, cujo domínio, juntamente com a habilidade de programar, constitui uma necessidade na lingüística computacional, é mais vantajosa na lingüística de corpus em comparação com programas comerciais e proprietários com interface gráfica.

Palavras-chave: sufixação, repetição, extração de dados, expressões regulares, ferramentas de linha de comando do Unix.

Introdução

No Brasil, a utilização de tecnologias da informação na investigação lingüística vem crescendo num ritmo consistente nos últimos 10 anos, o que se tem refletido na rápida expansão da lingüística de corpus[1] e, em menor medida, da lingüística computacional *stricto sensu*.[2] Visando a oferecer uma contribuição para a lingüística de corpus do português, a partir da perspectiva dessa segunda disciplina, com a qual está intimamente relacionada, este artigo explica, de forma detalhada, como utilizar ferramentas do sistema operacional UNIX para realizar pesquisas em corpora do português disponíveis livremente na WWW. As técnicas apresentadas serão exemplificadas por meio de sua aplicação no levantamento de dados lingüísticos em dois dos principais corpora eletrônicos do português que podem ser obtidos gratuitamente na Internet: (i) o Corpus Histórico do Português Tycho Brahe (doravante CHPTB), o maior e cronologicamente mais abrangente banco de dados digitalizado, e livremente acessível, de textos morfossintaticamente anotados do português e o

[1] Sobre esse ponto, ver Berber Sardinha e Almeida (2008) e os demais artigos reunidos no mesmo volume.

[2] Othero e Menuzzi (2005) e Othero (2006) atestam o incipiente interesse entre nós, por parte de lingüistas, no processamento automático da linguagem natural, área que, ao contrário do que se tem observado na Europa e nos Estados Unidos nas três últimas décadas, se restringiu no Brasil, durante muitos anos, quase que exclusivamente aos cursos de Informática, Ciência da Computação e Engenharia.

VEREDAS ON-LINE – LINGUÍSTICA DE CORPUS E COMPUTACIONAL – 2/2009, P. 134-150 – PPG LINGUÍSTICA/UFJF – JUIZ DE FOR A – ISSN 1982-2243

(ii) CETENFolha 1.0, também anotado, que reúne a totalidade das edições diárias do jornal Folha de São Paulo no ano de 1994, totalizando quase 24 milhões de palavras.[3]

1. A lingüística de corpus como área da informática aplicada à lingüística

Neste artigo, partimos do pressuposto de que o envolvimento com a lingüística de corpus, por parte de estudantes e pesquisadores nas áreas de Letras e Lingüística, será tão mais proveitoso quanto mais levar em conta o quadro interdisciplinar em que a disciplina está inserida, do qual faz parte também a lingüística computacional. Desse quadro, a informática é um componente fundamental. Conseqüentemente, todo pesquisador em lingüística de corpus que queira fazer jus à natureza interdisciplinar da sua área deve familiarizar-se com o uso do computador não apenas por meio de interfaces gráficas, mas também por meio de interfaces textuais, também chamadas interfaces de linha de comando. Neste ponto, antes de prosseguir, alguns esclarecimentos conceituais e terminológicos se fazem necessários.

A expressão *lingüística computacional* é polivalente, abrangendo quatro disciplinas diferentes, embora estreitamente relacionadas (AMTRUP, 2004, p. 2):

- LC1: *lingüística computacional* como subdisciplina lingüística voltada para os aspectos algorítmicos das línguas naturais e do processamento da linguagem natural;
- LC2: *lingüística computacional* como área de investigação direcionada ao desenvolvimento de ferramentas computacionais para a pesquisa lingüística e para o processamento de dados lingüísticos, correspondendo ao que Guinovart (2000) chama *informática aplicada à lingüística* ou *lingüística informática*;
- LC3: *lingüística computacional* como implementação de fenômenos da linguagem no computador, área conhecida também como processamento da linguagem natural (PLN), intimamente relacionada às áreas da inteligência artificial e da ciência da cognição de modo geral;
- LC4: *lingüística computacional* como ciência aplicada, voltada para o desenvolvimento de aplicativos para tradução automática, correção ortográfica e gramatical etc., constituindo um ramo da engenharia de software, como sugerem as designações em inglês *grammar engineering* ou *language technology*, respectivamente engenharia da gramática e tecnologia da linguagem.

O lingüista galego Xavier Gómez Guinovart, fundador do Seminário de Lingüística Informática da Universidade de Vigo, inclui a lingüística de corpus, que hoje não se concebe sem o uso do computador, na segunda das subáreas da lingüística computacional em sentido lato, a qual rotulamos como LC2 acima.

As disciplinas rotuladas como LC1 e LC3, por outro lado, constituem a lingüística computacional propriamente dita, correspondendo, grosso modo, ao que Guinovart (2000) denomina em galego "lingüística computacional por antonomasia". Essa disciplina visa, por um lado, a elaborar teorias lingüísticas computacionalmente implementáveis, utilizando o computador para a verificação automática de sua consistência e a comprovação de suas predições. Por outro lado, objetiva aplicar esses modelos na descrição de línguas naturais (GUINOVART, 2000).

Lemnitzer e Zinsmeister (2006, p. 9) definem a lingüística de corpus como atividade científica voltada para "a descrição de enunciados de línguas naturais, seus elementos e estruturas, e a elaboração teórica a partir dessa descrição, com base em análises de textos autênticos reunidos em corpora". Essa definição vai ao encontro da concepção de Santos (2008, p. 46-

[3] O CHPTB está disponível gratuitamente para acesso on-line ou *download* no seguinte endereço: <http://www.tycho.iel.unicamp.br/~tycho/corpus/>. O CETENFolha pode ser obtido no sítio da Linguateca: <http://www.linguateca.pt/CETENFolha/>.

47) de uma "linguística com corpos" (designação alternativa em português que propõe para *corpus linguistics*), a qual, segundo a autora, não tem nos corpora o seu objeto de estudo, considerados, antes, como utensílios para a investigação dos fenômenos da linguagem humana. Pelo menos em princípio, portanto, a disciplina independe da utilização do computador, não obstante a definição corrente de corpus como conjunto de dados lingüísticos digitalizados de natureza textual (SASAKI; WITT, 2004, p. 195).

De fato, a utilização de corpora na investigação de uma língua individual, de estágios históricos ou de dialetos de uma mesma língua ou mesmo de toda uma família de línguas é anterior ao advento da lingüística moderna, tendo constituído, por exemplo, a base da filologia clássica (PEDERSEN, 1972) e da gramática histórica do século XIX (JUNGEN; LOHNSTEIN, 2006).[4]

Graças à evolução tecnológica das últimas décadas, que barateou drasticamente os equipamentos de informática, a lingüística de corpus, ao mesmo tempo em que se tornou acessível a qualquer pesquisador que disponha de um computador e de programas especiais desenvolvidos para o processamento de corpora, passou, por outro lado, a não poder ser mais praticada sem esses recursos. Com efeito, numa disciplina de base fortemente empírica como a lingüística de corpus, em que amostras representativas de um fenômeno implicam a pesquisa num grande volume de textos, é impossível competir com o computador em velocidade e acuidade no levantamento de dados.[5]

Em sintonia com esses desdobramentos, assistimos no Brasil, nos últimos anos, a uma difusão cada vez maior da utilização de ferramentas computacionais na análise automática de textos com fins de extração de dados ou validação de hipóteses lingüísticas. Entre essas ferramentas, reina, nas universidades brasileiras, nos cursos de Letras ou Lingüística, quase absoluto, o programa WordSmith Tools (ver, por exemplo, Berber Sardinha, 2006).

Embora se caracterize por uma certa facilidade de uso, esse programa padece, contudo, de uma série de desvantagens, das quais destacamos as seguintes: (i) tem código *proprietário*, não permitindo a introdução de aperfeiçoamento por outros pesquisadores; (ii) exige licença que, para o usuário individual, custa 50 libras esterlinas; (iii) não é multiplataforma (só funciona em ambiente Windows).[6]

Neste artigo, pretendemos trilhar caminho oposto. Fazendo jus à natureza de informática aplicada da lingüística de corpus, desenvolveremos comandos e scripts em sistemas "livres", i.e. que possuem código aberto e se conformam aos princípios do chamado Projeto GNU da *Free Software Foundation*, sendo, ao mesmo tempo, gratuitos e multiplataforma.[7] Isso significa que funcionam tanto em sistemas operacionais comerciais como Windows e Mac OS X, quanto em sistemas gratuitos como o Linux. Outra vantagem do software livre é poder ser,

---

[4] Sobre as pesquisas precursoras da moderna lingüística de corpus, desde a Antiguidade, consulte-se, em português, o abrangente panorama traçado por Berber Sardinha (2000).

[5] Lemnitzer e Zinsmeister (2006) fazem um levantamento bastante abrangente de pesquisas recentes desenvolvidas no âmbito da lingüística de corpus do alemão, nos vários níveis de análise, cujos resultados somente foram viabilizados pela utilização do computador. Com isso, não estamos querendo dizer que uma análise totalmente automática seja possível para todos os fenômenos lingüísticos investigados por meio de corpora. Esses autores mesmos chamam atenção para casos em que uma intervenção humana na análise é imprescindível, dadas as limitações da tecnologia atualmente disponível para o processamento computacional de textos.

[6] É possível utilizar o WordSmith Tools, como outros programas do Windows, em outros sistemas operacionais, recorrendo a programas de virtualização como o CrossOver ou o WINE. Esses programas traduzem as instruções do sistema operacional original (no caso, o Windows) em instruções do sistema hospedeiro (por exemplo, Mac OS X ou Linux). Essa tradução, contudo, nem sempre é perfeita. Por outro lado, enquanto usuários do Linux dispõem do WINE, que é gratuito, usuários do Mac OS X dependem do CrossOver, que é pago.

[7] Sobre os critérios que um software deve satisfazer para ser considerado livre, consulte-se o sítio do Projeto GNU:< http://www.gnu.org/>.

devido a ter código aberto, modificado à vontade pelo usuário, que pode adaptá-lo a suas necessidades ou mesmo aperfeiçoá-lo.

Ao optar por essa via, seguimos o exemplo da lingüística de corpus na Alemanha, país onde predominam, concomitantemente à disseminação, há décadas, da lingüística computacional nos cursos de Lingüística, programas de código aberto e/ou gratuitos com interface de linha de comando para compilação e processamento de corpora. É significativo dessa situação que, em Lemnitzer e Zinsmeister (2006), o principal livro-texto na área escrito em alemão, o WordSmith Tools não seja sequer mencionado. Esse programa comercial é também ignorado por Ule e Hinrichs (2004, p. 236-237), que destacam, no âmbito do processamento automático de textos, por um lado, Perl, entre as linguagens de programação, e, por outro, as ferramentas de linha de comando do UNIX.

No Brasil, pelo contrário, a lingüística de corpus se ressente do distanciamento que ainda persiste entre as áreas de Letras, por um lado, e Computação e Informática, por outro. Isso se reflete, por exemplo, no fato de Berber Sardinha (2000, p. 334-335), ao mesmo tempo em que confere grande destaque ao WordSmith Tools, omitir a contribuição, em seu levantamento histórico dos avanços tecnológicos relacionados à compilação e ao processamento de corpora digitais, tanto das ferramentas de manipulação automática de textos do UNIX quanto da linguagem Perl. Contrariamente a Berber Sardinha (2004), que considera inviável a utilização, por parte do aluno comum de Letras, de programas de linha de comando, as seções seguintes mostram que esse tipo de interface, se explicado de forma suficientemente didática, dispensa conhecimentos prévios de programação.

Com este trabalho, esperamos contribuir para a redução dos custos das pesquisas lingüísticas, desonerando os orçamentos tanto de pesquisadores individuais, quanto das diferentes entidades governamentais de fomento à pesquisa, que não precisariam mais gastar com programas ou sistemas operacionais pagos como o Windows. É nossa intenção aqui, também, demonstrar a superioridade técnica e a maior flexibilidade de uso das ferramentas gratuitas e de código aberto que utilizam uma interface textual.

2. Em prol da lingüística de corpus no UNIX

Para realizar pesquisas em corpora, há alguns programas disponíveis que adotam uma interface gráfica, entre os quais se destaca no Brasil, como vimos, o WordSmith Tools. Na lingüística computacional (na acepção estrita definida acima), contudo, utilizam-se preferencialmente interfaces de linha de comando. Esse segundo tipo de interface confere muito maior rapidez às pesquisas e é muito mais flexível do que o primeiro tipo. Com uma rápida seqüência de comandos, podemos realizar de forma automática operações que demandariam muito mais tempo e esforço se levadas a cabo por meio de sucessivos movimentos e cliques de mouse. Limitamo-nos a três exemplos diretamente relevantes ao trabalho com corpora:[8]

- O corpus CETENFolha, na versão não anotada, é distribuído num único arquivo compactado que, uma vez expandido, ocupa 178 MB e contém 3.656.500 linhas. Por meio da ferramenta `split` do UNIX, podemos, num único comando, fragmentar esse arquivo em arquivos menores com um tamanho pré-determinado de bytes ou de linhas. Esses arquivos são nomeados automaticamente de uma forma sistemática (por exemplo, `CETEN_aa`,

[8] Mesmo com a alta sofisticação alcançada pelas atuais interfaces gráficas de sistemas operacionais como o Windows e o Mac OS X, a utilização das respectivas interfaces de linha de comando é vantajosa em muitas tarefas, sobretudo as de caráter repetitivo, constituindo, muitas vezes, a única maneira de explorar determinados recursos do computador, como atestam Knittel (2003) e Sobell e Seebach (2006).

`CETEN_ab`, `CETEN_ac`, `CETEN_ad` e assim por diante), o que permite recuperar a disposição de textos do original.

- Os 25 arquivos da versão morfossintaticamente anotada do CHPTB estão nomeados de tal forma que não podemos restringir cronologicamente uma pesquisa com facilidade. Por meio de um programa relativamente simples que elaboramos em UNIX, podemos renomear todos esses arquivos de forma automática, concatenando a data de nascimento do autor com o nome original do arquivo.[9] A partir do catálogo on-line de textos do CHPTB, esse programa transforma, por exemplo, o arquivo a_001_pos.txt.cs, que contém o texto *Reflexões sobre a Vaidade dos Homens*, de Matias Aires (nascido em 1705), em 1705_ a_001_pos.txt.cs.
- A versão anotada do CETENFolha adota um formato de representação vertical, em que cada palavra com sua anotação ocupa uma linha, o qual, segundo Ule e Hinrichs (2004, p. 226-227), é bastante vantajoso do ponto de vista do processamento computacional. O processo de dispor um texto nesse formato é chamado de toquenização, constituindo pré-requisito para a análise morfológica automática no paradigma de estados finitos (BEESLEY; KARTTUNEN, 2003, p. 315). Na seção 3, exploramos a utilidade de um programa em UNIX de nossa autoria, de apenas 4 linhas, capaz de criar, de uma só vez, uma versão toquenizada para cada um dos arquivos do CHPTB.

Outra vantagem dos programas com interface textual é que não fazem grandes exigências de hardware para funcionar a contento. Por exemplo, todos os comandos e scripts que elaboramos para o processamento de corpus expostos ao longo deste artigo são executados com rapidez também em computadores mais antigos ou de baixo custo.[10]

O preço que se tem a pagar por isso é a necessidade de aprender uma linguagem de programação, na qual se devem elaborar os diferentes comandos de que precisamos na manipulação de corpora. Na lingüística computacional, usou-se, nos últimos anos, sobretudo a linguagem Perl, acrônimo de "Practical Extraction and Report Language" – não por acaso criada por um dublê de programador e lingüista (SCHWARTZ; PHOENIX, 2001). Mais recentemente, essa linguagem tem sido paulatinamente substituída por Python, considerada, de modo geral, mais amigável, especialmente na aplicação à análise automática de textos, razão pela qual foi adotada pelo *Natural Language Toolkit* (NLTK), aparentemente o principal projeto atual na área de ensino do processamento da linguagem humana (BIRD; KLEIN; LOPER, 2009).

O sistema operacional UNIX, desenvolvido desde a década de 1970, que constitui o cérebro de máquinas do tipo PC equipadas com LINUX ou de computadores Macintosh (da empresa Apple) equipados com o Mac OS X, também oferece uma linguagem muito poderosa para se processar corpora, que é a linguagem do interpretador de comandos (chamado *shell* no jargão UNIX) `bash`. Por meio desse interpretador, temos acesso às facilidades oferecidas por poderosas ferramentas implementadas em UNIX, como os programas `grep`, `awk`[11] e `sed`, para manipulação de dados de textos.[12]

---

[9] Este programa será disponibilizado em breve no nosso sítio: <http://www.leonel.profusehost.net/>.

[10] Pudemos verificar isso executando esses comandos num Asus Eee PC 4G, um *netbook* com tela de 7" e menos de um quilograma, equipado com versão do Linux Xandros, que custou, em julho de 2008, apenas 240 Euros.

[11] Conforme Sobell e Seebach (2006, p. 609-610), o termo *awk* designa tanto uma linguagem para processamento de padrões quanto o programa ou ferramenta (*utility*) que executa comandos elaborados nessa linguagem.

[12] Usuários do Windows dispõem, atualmente, da possibilidade de utilizar o UNIX (especificamente o interpretador `bash` e as ferramentas de linha de comando mais comuns, inclusive aquelas mais úteis para a análise automática de textos) dentro do Windows, graças ao Projeto Cygwin, que simula o LINUX dentro do sistema operacional da Microsoft. Está disponível gratuitamente para *download* no seguinte endereço: <http://cygwin.com>.

A vantagem de se utilizar esses recursos tradicionais do UNIX em vez de recorrer à novidade representada por Python é o acesso direto a arquivos de textos e a maior rapidez na sua manipulação. Outro ponto forte em favor das ferramentas do UNIX é que, nas tarefas mais rotineiras de processamento de corpora, conseguimos elaborar comandos mais enxutos do que em Python. Sobretudo num contexto educacional, esse é um forte argumento em prol da utilização dessas ferramentas, tanto mais quanto Bird, Klein e Loper (2007, p. 363), ao defenderem Python como primeira linguagem de programação a ser aprendida para a análise automática de textos, ressaltam, ao lado da transparência, exatamente a simplicidade do código dessa linguagem em comparação com Perl, Prolog e Java, entre outras linguagens. Eles exemplificam isso com o programa em (1), que extrai, do arquivo de input standard, palavras terminadas em *ing* do tipo de *shopping* e *marketing*.[13]

(1)
```
import sys
for line in sys.stdin:
  for word in line.split():
    if word.endswith('ing'):
      print word
```

Para extrair palavras com esse sufixo de um fragmento do CETENFolha, contudo, precisamos de um programa mais longo em Python (ver (2)), que nomeamos `extrator.py` e podemos aplicar ao corpus por meio do comando em (3).

(2)
```
#! /usr/bin/env python
import sys
arquivo=open(sys.argv[1], 'rU')
for linha in arquivo.readlines():
  for palavra in linha.split():
    if palavra.endswith('ing'):
      print palavra
arquivo.close()
```

(3) `$ extrator.py CETEN_aa`

No UNIX, porém, alcançamos de forma mais veloz um melhor resultado com um comando bem mais curto do programa `grep`, de que trataremos na próxima seção:

(4) `$ grep -Ewo "[[:alpha:]]+ing" CETEN_aa`

Esse comando, ao contrário do programa (2), extrai também palavras seguidas de sinal de pontuação, como no exemplo (5) do CETENFolha, graças à opção `-w` (do inglês *word*).[14] O uso do `grep` é mais vantajoso também porque o aplicamos diretamente sobre o arquivo do corpus, ao passo que, em Python, temos de explicitamente abrir e fechar esse arquivo.

(5) `<s> A patinadora Tonya Harding, banida [...] . </s>`

O programa em Python ignora a palavra *Harding* nesse exemplo, pois está seguida de uma vírgula. Para corrigir esse problema mantendo a simplicidade e transparência características dessa linguagem, precisaríamos, por exemplo, introduzir em (2) disjunções lógicas para cada sinal de pontuação (por ex. “if palavra.endswith('ing') or palavra.endswith('ing,') or palavra.endswith('ing.')”). Uma alternativa mais econômica, porém menos transparente seria reformular essa condição como “if palavra.strip(string.punctuation).endswith('ing')” (importando o módulo `string`) ou recorrer, em um programa de nível maior de

[13] Para uma introdução ao mesmo tempo abrangente e acessível a Python, ver Chun (2006).

[14] O funcionamento da opção `-w` pode variar de uma variante do UNIX para outra, devido à inclusão ou não de caracteres com diacríticos entre os elementos constitutivos de uma “palavra”. Essa opção também é sensível ao tipo de codificação do texto.

complexidade, a expressões regulares. No `grep`, contudo, como vimos, obtemos o mesmo efeito, de forma muito mais prática, por meio da opção `-w`.[15]

Outro aspecto importante que deve ser levado em conta é que o UNIX tem uma posição bastante forte no mundo acadêmico nas áreas de computação e lingüística computacional. A familiaridade com a interface de linha de comando desse sistema é pré-requisito para se ter acesso a muitos projetos, sobretudo nas suas fases iniciais, quando ainda não se tem disponibilizadas interfaces gráficas ou versões para sistemas operacionais comerciais. No âmbito da lingüística de corpus, por exemplo, está implementado em UNIX um dos principais sistemas de processamemto de corpora, o *IMS Corpus Workbench* (CWB), do *Institute for Natural Language Processing* da Universidade de Stuttgart, na Alemanha. Esse sistema constitui a base do projeto AC/DC (Acesso a Corpora/Disponibilização de Corpora) da Linguateca. O UNIX subjaz também ao sistema de processamento de corpora paralelos *NATools*, desenvolvido por um dos pólos da Linguateca, o Departamento de Informática da Universidade do Minho, em Braga, Portugal.[16]

3. Estudo de caso: adjetivos em *-vel* no CHPTB

Com propriedade, Lemnitzer e Zinsmeister (2006, p. 130) consideram a formação de palavras como "a área mais criativa de uma língua cotidianamente utilizada". Num projeto pioneiro, os pesquisadores Lothar Lemnitzer e Tylman Ule, do Instituto de Lingüística da Universidade de Tübingen, na Alemanha, têm feito, desde o final do ano 2000, um levantamento diário das palavras novas surgidas em edições on-line de jornais alemães, registrando, em média, quinze novas criações lexicais por dia (LEMNITZER; ZINSMEISTER, 2006, p. 153).[17] Em português, a criatividade lingüística no campo do léxico pode ser constatada praticamente a cada edição de jornal e revista, quando nos deparamos com neologismos ainda não dicionarizados ou que não constam da nossa experiência lingüística anterior, mas que, mesmo assim, somos capazes de compreender.

Isso tem uma conseqüência fundamental para áreas tão diversas quanto o processamento automático da linguagem natural e o ensino de idiomas, entre outras: como o léxico da língua não é um conjunto fechado, podendo livremente ser expandido por meio da aplicação das regras de formação de palavras, é imprescindível levar em conta esse componente da morfologia num sistema de análise automática de textos ou num treinamento da habilidade de leitura.

Para descrever as regras de formação de palavras de uma língua como o português, precisamos não só inventariar os elementos formadores de palavras (raízes e afixos), mas também descrever as restrições de combinação entre esses elementos e as alterações morfofonológicas ou ortográficas resultantes. Outro aspecto fundamental é determinar a produtividade desses processos morfológicos, a qual permite determinar a aceitabilidade dos neologismos criados por meio deles, fornecendo, também, critérios para a elaboração de dicionários (LEMNITZER; ZINSMEISTER, 2006).

---

[15] No comando (4), a opção `-w` permite extrair também palavras com sinais de pontuação que não integram a classe de caracteres `string.punctuation` de Python (i.e. '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'), como é o caso do tipo de aspas duplas em exemplos do CETENFolha como «body piercing».

[16] Alguns dos principais ambientes de desenvolvimento de gramáticas computacionais estão implementados unicamente em UNIX, como é o caso, por exemplo, do TRALE, utilizado com o formalismo da HPSG (MÜLLER, 2007).

[17] A lista diariamente atualizada de novas criações lexicais, selecionadas de forma manual pelos pesquisadores como dignas de registro, a partir de uma listagem maior, compilada automaticamente, pode ser consultada no sítio do projeto: <http://www.wortwarte.de/>.

Como noutras áreas da lingüística, a investigação das regras de formação de palavras por meio da análise automática de corpora eletrônicos tem produzido resultados impossíveis de alcançar manualmente, como mostram Lemnitzer e Zinsmeister (2006). Nesta seção, exemplificamos como o UNIX pode ser útil nesse tipo de estudo, por meio de uma série de comandos e scripts que permitem explorar diversos aspectos do sufixo *-vel*, um dos mais produtivos da língua portuguesa, com base no CHPTB.

Segundo Anderson (1992), das regras de formação de adjetivos com o sufixo *-able* em inglês, *grosso modo* equivalente ao *-vel*, participam não só verbos, tanto transitivos quanto intransitivos, mas também nomes e mesmo "bases inexistentes" (por ex. em *possible* 'possível' e *affable* 'afável'). Uma investigação análoga sobre o sufixo *-vel* a partir do CHPTB deve começar naturalmente pelo levantamento das ocorrências de adjetivos com esse elemento, a fim de se poder catalogar e classificar as respectivas bases. Para tanto, construímos comandos em UNIX que levam em conta a maneira como o corpus está constituído e o tipo de anotação adotada, como veremos mais abaixo.

Na versão do CHPTB com o maior número de textos anotados, as palavras estão classificadas morfossintaticamente, recebendo uma etiqueta (*tag*) que indica tanto a sua categoria lexical (que se costuma designar em lingüística de corpus pela sigla POS, do inglês *part of speech*) quanto flexional, no caso das classes de palavras flexionáveis como substantivos, verbos etc.[18] Também os sinais de pontuação (entre os quais se incluem os parênteses) estão etiquetados. Em (6), temos um excerto das primeiras linhas do texto *Reflexões sobre a Vaidade dos Homens*, de Matias Aires (1705). A estrutura macrotextual é indicada por meio de etiquetas em XML. A classe de cada elemento (incluindo as próprias etiquetas em XML, anotadas como /CODE) é indicada por meio de anotação posicionada à sua direita, precedida de "/".[19] *En passant*, observemos que a etiqueta NPR, indicativa de substantivo (nome) próprio, é estranhamente aplicada às palavras *sobre* (preposição) e *vaidade* (nome comum).

(6)

```
<P_01>/CODE
<heading>/CODE REFLEXÕES/NPR-P SOBRE/NPR A/D-F VAIDADE/NPR DOS/P+D-P
HOMENS/NPR-P <_heading>/CODE
<P_02>/CODE
<P_03>/CODE <heading>/CODE Senhor/NPR :/. A-001,03.1/ID ./PONFP
<_heading>/CODE Ofereço/VB-P a/P Vossa/PRO$-F Majestade/NPR as/D-F-P
Reflexões/NPR-P sobre/P a/D-F vaidade/N dos/P+D-P homens/N-P ;/. A-
001,03.2/ID ./PONFP
```

Com base em listagem disponibilizada no sítio do CHPTB, definimos, para efeitos de consulta, por meio da expressão regular `[+\!\$.,\([:upper:]-]+`, uma língua formal que inclui não só o conjunto das etiquetas atualmente utilizadas nesse corpus, mas também outras que venham a ser criadas seguindo a mesma sintaxe do conjunto atual. Essa expressão caracteriza uma etiqueta morfossintática possível do CHPTB como sendo algo constituído de um ou mais caracteres da classe formada pelos sinais de adição e exclamação, cifrão, ponto, vírgula, parêntese à esquerda, letra maiúscula e hífen. No UNIX, podemos, por meio do comando em (7), armazenar essa noção numa variável T, cujo valor pode ser acessado por meio de `$T`. Esse comando pode também ser executado automaticamente pelo arquivo de configuração do usuário (`.profile` ou `.bash_profile`), de modo a tornar a variável disponível em cada sessão da interface de linha de comandos.

---

[18] Os organizadores do CHPTB consideram a etiquetagem em termos de classes de palavras como anotação morfológica, termo que, ao nosso ver, se aplicaria com mais propriedade a uma análise da estrutura interna das palavras, tal como no Corpus de Referência do Alemão, disponibilizado através do sistema COSMAS II no sítio <http://www.ids-mannheim.de/cosmas2/>. Preferimos seguir, nesse ponto, Lemnitzer e Zinsmeister (2006, p. 66), para quem as etiquetas de POS constituem anotação morfossintática.

[19] Para uma explicação detalhada sobre o sistema de edição e de anotação e o significado de cada sigla, consultar Sousa (2007) e Britto, Sousa e Galves (2008).

(7) $ **T=[+\!\$.,\([:upper:]-]+**

Outra noção importantíssima que podemos definir facilmente por meio de uma expressão regular é a de palavra da língua portuguesa, o que exemplificamos por meio do comando em (8), que armazena a expressão regular em questão na variável W. Essa expressão regular define língua formal de que as palavras da língua portuguesa, formadas a partir da concatenação de letras e do hífen, constituem um subconjunto.[20] Nessa expressão, utilizamos a classe de caracteres `[:alpha:]`, que denota qualquer letra do alfabeto.

(8) $ **W=[-[:alpha:]]+**

O comando em (9) invoca o programa `grep` (acrônimo de *global regular expression print*), a principal (e mais básica) ferramenta do UNIX para a lingüística de corpus, extraindo as palavras das duas primeiras linhas do texto transcrito parcialmente em (6).

(9) $ **grep -Eom 2 "$W/" a_001_pos.txt.cs**

```
REFLEXÕES/
SOBRE/
A/
VAIDADE/
DOS/
HOMENS/
Senhor/
```

O comando em (9) apresenta a estrutura típica de muitos comandos do UNIX, detalhada no Quadro 1. Nesse caso específico, as opções fornecem as seguintes instruções para o `grep`: (i) interpretar o padrão como expressão regular estendida (`-E`), (ii) apresentar apenas as ocorrências (`-o`) e (iii) limitar os resultados ao máximo de duas linhas (`-m 2`).[21]

Nome do comando	Opções	Padrão	Lista de arquivos
grep	-Eom 2	$W/	a_001_pos.txt.cs

Quadro 1: Estrutura do comando (9).

Nesse comando, incluímos na expressão regular a barra inclinada à direita, de modo a poder extrair apenas formas da língua portuguesa, excluindo tudo que se refere à metalinguagem utilizada pelo CHPTB, que ocorre à direita da barra. Para remover a barra inclinada dos resultados, é só canalizá-los para o programa `awk` como no comando em (10), definindo, por meio da opção `-F`, a barra inclinada à direita como separador entre campos. Nesse caso, definimos o espaço em branco como separador entre registros (ORS, do inglês *output register separator*), o que faz os resultados serem apresentados não numa coluna, mas horizontalmente. O programa `awk`, que processa comandos na linguagem de processamento de padrões homônima, é outra das ferramentas do UNIX mais úteis na lingüística de corpus.

(10) $ **grep -Eom 3 "$W/" a_001_pos.txt.cs | awk -F/ -v "ORS= " '{print $1} END {print "\n"}'**

```
REFLEXÕES SOBRE A VAIDADE DOS HOMENS Senhor Ofereço a Vossa Majestade as Reflexões sobre a vaidade dos homens
```

No comando acima temos o que se chama em UNIX de “cano” (*pipe*). Por meio do operador “|” (barra vertical), direcionamos o output de um programa para processamento por outro (no caso, respectivamente, o `grep` e o `awk`), o qual atua como espécie de filtro.

Por meio das expressões regulares armazenadas nas variáveis T e W podemos, sem muito esforço, fazer um levantamento, na versão do CHPTB anotada morfossintaticamente, da

[20] Essa expressão regular é insuficiente para definir as palavras de línguas como o alemão, em que se podem construir palavras a partir de algarismos como o substantivo *80er-Jahre* ‘os anos 80’ ou o adjetivo *8-jährig* ‘de oito anos’.

[21] A opção `-m 2` não limita a pesquisa às duas primeiras linhas do texto; em vez disso, exibe as ocorrências das duas primeiras linhas onde há pelo menos uma ocorrência em cada uma.

freqüência com que cada etiqueta de classificação de palavra (excluindo os sinais de pontuação) foi utilizada. O comando em (11) exibe as 5 etiquetas mais utilizadas. A opção `-h` do `grep` permite realizar a pesquisa em todos os textos do corpus como se estivessem reunidos num único arquivo. Como podemos constatar, N e P, respectivamente substantivo comum (i.e. não próprio) no singular e preposição não amalgamada com artigo, são as duas mais freqüentes. Nesse levantamento, a classificação da categoria NPR em terceiro lugar deve ser considerada com cautela, diante da possibilidade de erro de anotação que apontamos acima.

(11)
```
$ grep -Eoh "$W/$T" *pos.txt.cs|awk -F/ '{print $2}'|sort|uniq -c
| sort -nr | head -5
 127655 N
 108893 P
  64816 NPR
  58110 CONJ
  51851 N-P
```

No comando acima, primeiro o `grep` extrai todas as palavras com suas respectivas anotações. O `awk` é responsável pela extração das etiquetas associadas a palavras. Em seguida, processamos o resultado desse comando com a ferramenta `sort`, que o organiza em ordem alfabética. Em mais uma etapa de processamento, fazemos uma contagem das linhas contíguas repetidas por meio do comando `uniq -c`. Finalmente, ordenamos de forma numericamente decrescente os resultados por meio da opção `-nr`, que instrui a ferramenta `sort` a realizar um ordenamento pelo critério numérico na ordem reversa à default, e filtramos com o comando `head` os 5 primeiros resultados.

Essa última combinação de 4 comandos é tão útil na lingüística de corpus que vale a pena incluí-la num script (ver (12)), como são chamados os programas criados pelo usuário na linguagem da interface de linha de comandos `bash`.[22] Supondo que esse script foi salvo com o nome `conta` num diretório especificado na variável de ambiente `PATH` e tem permissão de execução, podemos utilizá-lo para contar ocorrências canalizando para ele o output de um outro programa, como veremos mais abaixo.

(12)
```
#!/bin/bash
nome=$(basename $0); : ${TEMPDIR:=/tmp}
tmp_out=$TEMPDIR/$$.${nome}1; tmp_in=$TEMPDIR/$$.${nome}2
exec 3<&0; cat <&3 > $tmp_in; exec 3<&-
sort $tmp_in |uniq -c| sort -k 1,1nr -k 2f > $tmp_out
if [ $# -eq 1 ]; then
        head -n $1 $tmp_out; else
        set $(wc -l $tmp_in)
        awk -v "total=$1" < $tmp_out '
        {printf "%-25s%-25d%5.4f%%\n",$2,$1,(100/total)*$1}
        END {printf "\n\n\t\tTotal:%d\n", total}
'
fi
```

As duas expressões regulares definidas em (7) e (8) são igualmente úteis para extrairmos as ocorrências de um determinado tipo de palavra, como verbos com infinitivo flexionado ou adjetivos no superlativo, só para dar dois exemplos. Retomando a questão da formação de palavras por meio da sufixação com *-vel*, podemos fazer um levantamento das formas de adjetivos com essa terminação, no singular ou no plural, ocorrentes nos 25 textos anotados morfossintaticamente do CHPTB, por meio de variante do comando em (13) sem a parte `|head -5`, que limita a exibição de resultados ao máximo de 5 ocorrências. O operador `?`

[22] Nesse script, aperfeiçoamos a ordenação dos resultados, que é agora numericamente decrescente com relação à freqüência, mas alfabeticamente crescente para as palavras, no caso de itens com a mesma freqüência.

indica que a expressão regular imediatamente precedente (no caso, o valor da variável T) é facultativa.

(13) `$ grep -Eoh "${W}ve(l|is)/ADJ$T?" *pos.txt.cs | head -5`

```
impossíveis/ADJ-G-P
memorável/ADJ-G
repreensível/ADJ-G
admiráveis/ADJ-G-P
agradável/ADJ-G
```

Quantas formas (*tokens*) de adjetivos em *-vel* ocorrem no CHPTB? O comando em (14) nos fornece esse dado. Nesse caso, canalizamos a lista de formas produzida pelo `grep` para a ferramenta `wc`, que, com a opção `-l`, conta as linhas de um arquivo.

(14) `$ grep -Eoh "${W}ve(l|is)/ADJ$T?" *pos.txt.cs | wc -l`

```
2170
```

Nos comandos acima, extraímos apenas as formas de adjetivos em *-vel*, desprezando todo o seu entorno nas sentenças do corpus. Uma análise da formação desses adjetivos, contudo, não pode prescindir de uma análise do contexto. Para produzir no UNIX concordâncias num formato análogo ao consagrado KWIC (*keyword in context*), o mais prático é toquenizar os arquivos a serem pesquisados, o que o script em (15) é capaz de fazer, em segundos, com todos 25 os arquivos morfossintaticamente anotados do CHPTB:

(15)

```
#! /bin/bash
for i in *_pos.txt.cs; do
  set $(echo "$i" | awk 'BEGIN {FS= "."} { print $1}')
  tr -s '[:blank:]' '\n' < "$i" > "$1".tok.txt; done
```

O primeiro comando abaixo define o alias `grec`, que consiste numa versão do `grep` com as opções `-E` e `--color=always`, por meio da qual as concordâncias são destacadas em cor. O segundo comando exibe, separadas por --, as duas primeiras ocorrências de adjetivo em *-vel* na versão toquenizada do texto de Matias Aires, com um contexto de dois elementos (palavras, sinais de pontuação etc.) acima e abaixo (i.e. à esquerda e à direita, respectivamente, na versão não toquenizada).

(16) `$ alias grec='grep -E --color=always'`

(17) `$ grec -m 2 -C 2 ".+vel/ADJ.*" a_001_pos.tok.txt`

```
fáceis/ADJ-G-P
aqueles/D-P
impossíveis/ADJ-G-P
./.
A-001,04.16/ID
--
certa/ADJ-F
aquela/D-F
memorável/ADJ-G
profecia/N
,/,
```

Outra estratégia de extrair concordâncias é definir o contexto por meio de uma expressão regular, armazenando-a numa variável, como em (18). O comando em (19) exemplifica como obter as mesmas concordâncias de (17) por esse método, mas agora com um contexto de até três elementos à esquerda e à direita. Com a opção `-n`, o `grep` numera as linhas (41 e 89, no caso). Esse comando evidencia o potencial do `grep` para a extração não apenas de palavras individuais, mas também de colocações e padrões sintáticos como, por exemplo, a colocação pós-nominal ou pré-nominal de adjetivos.

(18) `$ C=[^[:space:]]+`

(19) `$ grep -Enom 2 "($C){0,3}${W}ve(l|is)/ADJ$T?($C){0,3}" a_001_pos.txt.cs`

```
41:e/CONJ fáceis/ADJ-G-P aqueles/D-P impossíveis/ADJ-G-P ./. A-001,04.16/ID ./PONFP
```

```
89:é/SR-P certa/ADJ-F aquela/D-F memorável/ADJ-G profecia/N ,/, que/WPRO
```

Uma limitação do CHPTB é não estar lematizado, contrariamente ao que ocorre com o CETENFolha. Dessa forma, não podemos, no primeiro corpus, obter diretamente os tipos (*types*) de adjetivos em *-vel*, com as respectivas quantidades de ocorrências (*tokens*). Essa informação tem uma importância lingüística muito grande, na medida em que permite identificar as bases mais produtivas para a formação de adjetivos com esse sufixo. No comando abaixo, extraímos as três primeiras ocorrências de adjetivos em *-vel* no segundo corpus, realizando uma busca pelos lemas terminados por esse sufixo e categorizados como ADJ.

(20) $ **grep -Em 3 "vel][[:space:]]ADJ" cetenfolha1.0.cg**
```
factíveis       [factível] ADJ F P @N<
terríveis       [terrível] ADJ F P @N<
impossível      [impossível] ADJ M S @<SC
```

Graças à lematização, a contagem dos tipos de adjetivos em *-vel* no CETENFolha pode ser feita por meio de uma seqüência relativamente simples de comandos. Primeiro, extraímos com o `awk` apenas os lemas e utilizamos a ferramenta `tr` para remover os colchetes, enviando o resultado para o arquivo `vel.txt` (ver (21)). Em seguida, canalizamos, por meio da ferramenta `cat`, o conteúdo desse arquivo para o script `conta`, que realiza uma contagem dos tipos de adjetivos, exibindo os 5 mais freqüentes (ver (22)).

(21) $ **awk '$2 ~ /vel]$/ && $3 ~ /ADJ/ {print $2}' cetenfolha1.0.cg | tr -d [\]\[] > vel.txt**

(22) **$ cat vel.txt| conta 5**
```
6779 possível
3702 responsável
1563 favorável
1373 impossível
1220 disponível
```

No UNIX, dispomos de ferramentas por meio das quais podemos, com relativa facilidade, realizar uma lematizacão dos adjetivos em *-vel* no CHPTB, como mostraremos na seção 4.

Uma primeira aproximação para o problema do cálculo das bases mais produtivas no CHPTB é canalizar o resultado do comando `grep` de (14) para a ferramenta `tr`, capaz de realizar substituições de caracteres em um determinado input. No caso, precisamos substituir maiúsculas por minúsculas, de modo a tratar cadeias do tipo de *Miserável* e *miserável* como *tokens* de um mesmo tipo. Em (23), criamos um chamado *alias*, para evitar digitar um comando muito longo em (24). Nesse comando, utilizamos o script `conta` de (12) como filtro, exibindo as dez formas mais freqüentes. Para exibir uma quantidade diferente de formas, basta especificar o número desejado logo após o nome do script.

(23) $ **alias adj='grep -Eoh "${W}ve(l|is)/ADJ$T?" *pos.txt.cs'**

(24) $ **adj|tr "[[:upper:]]" "[[:lower:]]"| conta 10**
```
189 possível/adj-g
103 impossível/adj-g
75 miserável/adj-g
61 notável/adj-g
59 agradável/adj-g
53 admirável/adj-g
34 amável/adj-g
29 miseráveis/adj-g-p
28 venerável/adj-g
28 veneravel/adj-g
```

O resultado do comando (24) expõe um outro problema do CHPTB: a modernização da grafia, adotada como norma geral na edição dos textos (com poucas exceções previstas), de modo a facilitar a análise lingüística automática (SOUSA, 2007, p. 41-43), não é sempre consistente, pois, como se pode constatar acima, grafa-se *venerável* ora com, ora sem acento.

Entre as *hapax legomena* que obtemos com a variante (25) do comando (24), encontramos evidentes lapsos, como *agrdável/adj-g*, *amaveis/adj-g-p*, *inhabitavel/adj-g*, *inmovel/adj-g*, *horriveis/adj-g-p*, *soportaveis/adj-g-p* e *spunhavel/adj-g*. Nesse comando, utilizamos o `awk` para extrair as ocorrências com freqüência igual a 1.

(25) `$ adj |tr "[[:upper:]]" "[[:lower:]]"|sort|uniq -c|awk '$1 == 1'`

Uma maneira mais sofisticada e elegante, pois não minusculiza as etiquetas como em (24), de colocar todos os adjetivos em minúsculas e, em seguida, contá-los, está em (26) e (27). Em (26), utilizamos o `awk` para minusculizar apenas os adjetivos, que ocupam o primeiro campo (referenciado por meio de $1) de cada linha de texto, redirecionando o resultado desse processamento para o arquivo `out`. Em (27), o script `conta` é invocado sem nenhum argumento. Nesse caso, ele realiza uma contagem das formas e exibe a freqüência de cada uma em termos percentuais em relação ao total de adjetivos em *-vel*.

(26) `$ adj | awk -F/ {print tolower($1) "/" $2 }' > out`

(27) `$ cat out | conta`

```
possível/ADJ-G          189                     8.71%
impossível/ADJ-G        103                     4.75%
miserável/ADJ-G         75                      3.46%
notável/ADJ-G           61                      2.81%
agradável/ADJ-G         59                      2.72%
...

                Total:2170
```

4. Explorando o poder das memórias de expressões regulares

Um dos recursos mais poderosos do formalismo das expressões regulares do UNIX consiste na capacidade que tanto o `grep` quanto outras ferramentas desse sistema operacional têm de memorizar uma cadeia (*string*) que satisfaz determinado padrão (definido por uma expressão regular) e reutilizar o que foi memorizado para, por exemplo, encontrar repetições dessa concordância num determinado segmento de texto. Esse recurso, de que também dispõe, por exemplo, a linguagem Perl (SCHWARTZ; PHOENIX, 2001, p. 109-11), não está disponível em programas de interface gráfica como o Wordsmith Tools (SCOTT, 2009).

As memórias das expressões regulares constituem um valioso instrumento no estudo de fenômenos nos vários níveis de análise lingüística, da fonologia e grafêmica à estilística, passando pela sintaxe. Por exemplo, uma das questões a ser investigadas na descrição da ortografia do português em documentos históricos, conforme Farias e Ximenes (2008), refere-se ao emprego de "consoantes duplas" e "vogais dobradas". Em outras palavras, um lingüista que se debruce sobre isso deverá responder à seguinte pergunta: Que letras se repetem de forma adjacente no corpus (i.e. quais são as seqüências do tipo *aa*, *ss* etc.)?

Para investigar essa questão, fragmentamos inicialmente, com o comando em (28), a versão não anotada do corpus CETENFolha em blocos de 30 MB. Em seguida, construímos uma expressão regular com uma memória que utilizamos no comando do `grep` em (29), o qual exibe as 10 primeiras concordâncias no arquivo `CETEN_aa`, que contém o primeiro bloco de 30 MB:

(28) `$ split -b 30MB CETENFolha-1.0 CETEN_`

```
CETEN_aa        CETEN_ac        CETEN_ae
CETEN_ab        CETEN_ad        CETEN_af
```

(29) `$ grep -Eom 10 '[-[:alpha:]]*([[:alpha:]])\1[[:alpha:]]*' CETEN_aa | awk -v "ORS= " '{print $1}'`

```
supreendente muitíssimo Confissões Confissões teens disso assinada
impressora Free Esse sucesso assessor disse assunto
```

Esses resultados não são muito interessantes, uma vez que a repetição das letras *s* e *r* e das vogais *e* e *o* constituem os casos mais triviais. Na versão seguinte do comando, refinamos a

busca para extrair apenas repetições não triviais, obtendo apenas nomes próprios, siglas ou estrangeirismos:

(30) $ **grep -Eom 15 '[-[:alpha:]]*([a-df-npqt-zA-DF-NPQT-Z])\1[[:alpha:]]*' CETEN_aa | awk -v "ORS= " '{print $1}'**
Caparelli Applause Schwarzenegger Schwarzenegger LANCELLOTTI yuppie Ccocco EXXON Exxon Exxon Campitti Unplugged BBC Belinatti Benetton

Um outro exemplo de aplicação das memórias de expressões regulares é na investigação da repetição de palavras como recurso retórico em português. Que classes de palavras e em que tipos de estruturas são iteradas? Qual o efeito semântico dessas iterações? O comando em (31) fornece de forma rápida dados, dos quais transcrevemos alguns dos mais interessantes, que podem subsidiar a elucidação dessas questões. As variáveis `N` e `S` armazenam, respectivamente, as expressões regulares `[^-[:alpha:]]` (o conjunto complementar da expressão armazenada na variável `W`, definida em (8)) e `[[:space:]]+` (um ou mais espaços em branco).

(31) $ **grep --color=always -En "$N($W)(,$S\1)+$N" CETEN_aa**
63177:<s> Se não enxotar o Fernando Henrique, ele vai continuar **falando**, **falando**, ameaçando, e não sai do PMDB </s>
193752:<s> Assim, eu temo que seja **muito**, **muito**, **muito** atual para a política britânica, porque é o que nós fazemos o tempo todo (ri) . </s>
265613:<s> E **alô**, **alô** seu Zagalo, que tal estrear a nova camisa amarelinha das quatro estrelas com os cinco? </s>
383580:<s> «A gente **espera**, **espera** e eles nunca vêm . </s>
402221:<s> Eles são **sujos**, **sujos**, **sujos**." </s>

Para concluir a presente seção, mostramos como utilizar essa técnica do UNIX para fazer um levantamento dos lemas dos adjetivos terminados em *-vel*.

O comando (32), que utiliza o programa `sed` do UNIX (um editor de texto não interativo), substitui as formas do plural pelas correspondentes formas do singular no arquivo `out`, criado com o comando (26), armazenando os resultados no arquivo `lemas`. Para tanto, o programa extrai o radical de cada adjetivo e o armazena na memória \1. Utilizamos a memória \3 para extrair a etiqueta morfológica. Por meio da ferramenta `head`, em (33), exibimos as cinco primeiras linhas do arquivo `lemas`. Note que, com a lematização, a etiqueta ADJ-G-P, onde P indica plural, é precedida das formas de citação dos adjetivos, no singular.

(32) $ **sed -r "s@($W)ve(l|is)(/ADJ$T?)@\1vel\3@g" out > lemas**
(33) $ **head -5 lemas**
impossível/ADJ-G-P
memorável/ADJ-G
repreensível/ADJ-G
admirável/ADJ-G-P
agradável/ADJ-G

No comando (34), processamos o arquivo `lemas` com o `awk`, de modo a abstrair da anotação morfossintática, canalizando o resultado para o nosso script `conta`, que exibe os 5 lemas mais freqüentes, diferentemente de (27), que contabiliza apenas as formas dos adjetivos. Note que as 104 ocorrências do lema *miserável* resultam da soma de 75 formas no singular e 29 no plural.

(34) $ **awk -F/ '{print $1}' lemas | conta 5**
205 possível
113 impossível
104 miserável
76 notável
73 agradável

Conclusão

A extração de dados de corpora eletrônicos como meio de investigação de fenômenos da linguagem constitui uma das atividades fulcrais da lingüística de corpus. Neste trabalho, tratamos dessa questão sob o enfoque da lingüística computacional, disciplina irmã dentro do quadro interdisciplinar da informática aplicada. Uma conseqüência inevitável dessa abordagem é a exploração de corpora por meio de ferramentas de interface textual, em que os comandos são formulados numa determinada linguagem de programação.

Na lingüística de corpus, a linguagem de programação Perl constituiu durante bastante tempo quase um padrão, mas tem recentemente cedido lugar, cada vez mais, a Python. Essas duas linguagens, de uso totalmente gratuito, oferecem aos pesquisadores a possibilidade de explorar todo o poder das expressões regulares. Mais importante, como linguagens de programação geral, permitem implementar algoritmos capazes de automatizar tarefas para qualquer problema computável. Programas de interface gráfica comerciais como o WordSmith Tools, muito popular nos cursos de Letras e Lingüística no Brasil, não têm essa capacidade.

Este artigo teve como objetivo central mostrar as vantagens da utilização das ferramentas para processamento de textos do UNIX e da linguagem da interface de linha de comando desse sistema operacional. Também frutos do movimento do software livre, esses programas permitem construir comandos mais enxutos do que em Perl e Python para muitas das principais tarefas de processamento automático de textos. Para exemplificar isso, elaboramos comandos e scripts em UNIX capazes de extrair dados de natureza tanto qualitativa como quantitativa para investigações empíricas sobre a língua portuguesa em duas áreas, a partir de corpora livremente disponíveis para *download* na Internet: o Corpus Histórico do Português Tycho Brahe e o CETENFolha 1.0. A primeira área de aplicação dessas técnicas foi a morfologia derivacional, especificamente os adjetivos sufixados com *-vel*. Nesse estudo de caso, pudemos constatar inconsistências na editoração e anotação do primeiro corpus, ao mesmo tempo em que mostramos como superar uma grave limitação sua, que é a falta de lematização. A segunda área enfocada foi a ocorrência de repetições de unidades lingüísticas nos níveis grafêmico e lexical, área de investigação que releva ao estudo tanto da ortografia quanto da sintaxe e da estilística.

Abstract: This paper approaches corpus linguistics as a subfield in applied informatics which features among its main focuses automatic data extraction from corpora. For this purpose, we develop commands and scripts in the UNIX bash command language, illustrating its applicability in the investigation of the *-vel* suffix and of iterations of letters and words in two of the main corpuses of Portuguese. We argue that using free software tools with textual interface, whose mastering together with programming skills is a necessity in computational linguistics, is more advantageous in corpus linguistics in comparison to commercial and proprietary programs with graphical interface.

Keywords: suffixation, iteration, data extraction, regular expressions, Unix command line tools.

Referências

AMTRUP, J. W. Aspekte der Computerlinguistik. In: KLABUNDE, R. et al. (Eds.). *Computerlinguistik und Sprachtechnologie:* eine Einführung. Heidelberg: Spektrum Akademischer Verlag, 2004. p.1-15.

ANDERSON, S. R. *A-morphous morphology*. Cambridge: Cambridge University Press, 1992. 434p.

BEESLEY, K. R.; KARTTUNEN, L. *Finite state morphology*. Stanford: CSLI Publications, 2003. 510p.

BERBER SARDINHA, T. Lingüística de Corpus: histórico e problemática. *D.E.L.T.A.*, São Paulo, v. 16, n. 2, p. 323-367, 2000.
BERBER SARDINHA, T. Lingüística de Corpus: uma entrevista com Tony Berber Sardinha. *ReVEL*, v. 2, n. 3, 2004. Disponível em:< http://www.revel.inf.br/> Acesso em: 2 set. 2009.
BERBER SARDINHA, T. Preparação de material didático para Aprendizagem Baseada em Tarefas com WordSmith Tools e corpora. *Calidoscópio*, São Leopoldo, v. 4, n. 3, p. 148-155, 2006.
BERBER SARDINHA, T.; ALMEIDA, G. M. B. A Linguística de *Corpus* no Brasil. In: TAGNIN, S. E. O.; VALE, O. A. (Eds.). *Avanços da Linguística de* Corpus *no Brasil*. São Paulo: Humanitas, 2008. p.17-40.
BIRD, S.; KLEIN, E.; LOPER, E. *Natural language processing with Python*: analyzing text with the Natural Language Toolkit. Sebastopol, CA: O'Reilly, 2009. 502p. Disponível em: <http://www.nltk.org/book> Acesso em: 25 abr. 2009.
BIRD, S.; KLEIN, E.; LOPER, E. *Introduction to natural language processing*. 2007. 382p. Disponível em: <http://www.nltk.org/book> Acesso em: 14 mai. 2007.
BRITTO, H.; SOUSA, M. C. P.; GALVES, C. *The Tycho Brahe Corpus annotation system*: morphological tags. Campinas: IEL/UNICAMP, 2008. Disponível em:<http://www.tycho.iel. unicamp.br/ ~tycho/corpus/ manual/ tags.html> Acesso em: 9 abr. 2009.
CHUN, W. J. *Core Python programming*. 2. ed. Upper Saddle River, NJ: Prentice Hall, 2006.
FARIAS, E. M. P.; XIMENES, E. E. A ortografia no Brasil e em Portugal: práticas de escrita em documentos do século XVII. *Revista Portuguesa de Humanidades*, Braga, 2009. No prelo.
GUINOVART, X. G. Lingüística computacional. In: RAMALLO, F.; REI-DOVAL, G.; RODRÍGUEZ, X. P. (Eds.). *Manual de ciencias da linguaxe*. Vigo: Xerais, 2000. p. 221-268.
JUNGEN, O.; LOHNSTEIN, H. *Einführung in die Grammatiktheorie*. München: Wilhelm Fink, 2006. 165p.
KNITTEL, B. *Windows XP under the hood*: hardcore Windows scripting and command line power. Indianapolis: Que, 2003. 722p.
LEMNITZER, L.; ZINSMEISTER, H. *Korpuslinguistik*: eine Einführung. Tübingen: Narr, 2006. 220p.
MÜLLER, S. *Head-Driven Phrase Structure Grammar*: eine Einführung. Tübingen: Stauffenburg, 2007. 440p.
OTHERO, G. A. *Teoria X-barra:* descrição do português e aplicação computacional. São Paulo: Contexto, 2006. 160p.
OTHERO, G. A.; MENUZZI, S. M. *Lingüística computacional*: teoria & prática. São Paulo: Parábola, 2005. 128p.
PEDERSEN, H. *The discovery of language*: linguistic science in the 19th century. Bloomington; London: Indiana University Press, 1972. 360p.
SANTOS. D. *Corporizando algumas questõe*s. In: TAGNIN, S. E. O.; VALE, O. A. (Eds.). *Avanços da Lingüística de* Corpus *no Brasil*. São Paulo: Humanitas, 2008. p.41-66.
SASAKI, F.; WITT, A. Linguistische Korpora. In: LOBIN, H.; LEMNITZER, L. (Eds.). *Texttechnologie*: Perspektiven und Anwendungen. Tübingen: Stauffenburg, 2004. p. 195-216.
SCHWARTZ, R. L.; PHOENIX, T. *Learning Perl*. 3. ed. Sebastopol, CA: O'Reilly, 2001. 316p.
SCOTT, M. *Step-by-step guide to Wordsmith*. Disponível em: <http://www.lexically.net/ wordsmith/step_by_step/index.html> Acesso em: 14 abr. 2009.
SOBELL, M. G.; SEEBACH, P. *A practical guide to UNIX for Mac OS X users*. Upper Saddle River, NJ: Prentice Hall, 2006. 999p.
SOUSA, M. C. P. *Sistema de Edições Eletrônicas do Corpus Histórico do Português Tycho Brahe*: Fundamentos, Diretrizes e Procedimentos. Campinas: IEL/UNICAMP, 2007. 60p.

Disponível em:< http://www.tycho.iel.unicamp.br/ ~tycho/ corpus/ manual/prep/index.html> Acesso em: 21 abr. 2009.
ULE, T.; HINRICHS, E. Linguistische Annotation. In: LOBIN, H.; LEMNITZER, L. (Eds.). *Texttechnologie*: Perspektiven und Anwendungen. Tübingen: Stauffenburg, 2004. p. 217-243.