

UNIVERSIDADE DO MINHO

MESTRADO INTEGRADO EM ENGENHARIA
INFORMÁTICA

Processamento de Linguagens

Filipe Monteiro (a80229)

Bruno Martins (a80410)

28 de Abril de 2019

Resumo

Este trabalho tem como foco a utilização da ferramenta *GAWK* lado a lado com expressões regulares para processamento de *Corpus Linguístico* (neste caso apresentados em formato *Freeling*) e recolha de informações destes. Este relatório descreve uma série de programas que recolhem diferentes tipos de informação dos documentos, entre eles, por exemplo, a recolha dos nomes de todos as personagens de *Harry Potter*, ou ainda, calcular a lista de verbos, substantivos e outros, apresentando o resultado numa página HTML.

Conteúdo

1	Introdução	3
2	Conceitos Base	3
2.1	Formato do Documento	3
2.2	Estruturas de dados	4
3	Análise e Solução dos exercícios	5
3.1	Ex A - Número de extratos	5
3.2	Ex B - Personagens do Harry Potter	7
3.3	Ex C - Listar Substantivos, Verbos, Adjectivos, Advérbios num ficheiro <i>HTML</i>	9
3.4	Ex D - Determinar o dicionário implícito no <i>córpore</i>	11
4	Conclusão	12

1 Introdução

Este projeto consiste na utilização da ferramenta *GAWK* para o processamento de textos no formato *Freeling* de forma a criar respostas apropriadas ao requisitado. Para o aprimoramento das questões respondidas também foram utilizadas expressões regulares. As questões propostas variavam desde encontrar nomes próprios das personagens dos livros *Harry Potter*, contar o número de extratos existentes num ficheiro e criar um pequeno dicionário de palavras. Nas secções seguintes é apresentado o caso de estudo bem como todas as decisões que foram tomadas para criar os filtros necessários.

2 Conceitos Base

Para a total compreensão deste relatório é necessário o conhecimento prévio de alguns conceitos. Numa primeira instância Expressão Regular que são utilizadas para a verificação de alguns padrões existentes. *GAWK* que é uma linguagem desenhada para o processamento de dados e, também é utilizada como forma de extração de informação de ficheiros. Estruturas de dados, são formas de armazenar informação de forma a manipula-la como pretendemos. *FreeLing*, é uma biblioteca de análise de linguagens, que permite dividir frases e fazer análise morfológica, poderemos ver um exemplo na subsecção seguinte.

2.1 Formato do Documento

Para melhor entendimento do documento fornecido, algumas palavras-chave são necessárias:

- Extratos: conjunto de *records*, separados por 1 linha em branco, contendo várias linhas numeradas de 1 a N, que correspondem às palavras pertencentes a esse extrato;
- *Record*: linha contendo a informação sobre 1 palavra.

Os *corpora* fornecidos, em formato *Freeling*, possuem extractos, onde cada *record* possui várias colunas, das quais apenas algumas são relevantes:

- num: número o *record*;
- palavra: palavra a que se refere ao *record*;
- lema: forma gráfica de uma palavra que é usada como entrada de verbete em dicionários ou vocabulários;

- *pos(part of speech)*: classe morfológica da palavra;
- *features*: papel da palavra na frase onde se encontra;
- *árvore*: contém informação adicional sobre a palavra.

2.2 Estruturas de dados

Para a realização deste projeto a única estrutura de dados utilizados foram *arrays*. Devido ao facto da linguagem *GAWK* ter arrays dinâmicos já incorporados isto permite uma maior versatilidade na forma como armazenamos a informação proveniente dos extratos processados. Foram numa primeira instância utilizados arrays unidimensionais que apenas permitem o armazenamento contíguo de dados, mas também foram utilizados *arrays* multidimensionais de forma a conseguir armazenar informação relacionada entre si.

3 Análise e Solução dos exercícios

Como cada desafio tinha um objetivo diferente, iremos explicar a nossa análise e resolução mediante esta mesma.

3.1 Ex A - Número de extratos

Neste exercício, é pretendido recolher o número de extratos existentes no *corpora*. Ao verificarmos os documentos fornecidos, para além de sabermos que os extratos estavam divididos por linhas em branco, concluímos que cada extrato tinha um *record* com o número 1 (no índice 1 das colunas), logo poderíamos contar o número de vezes que este aparecia ao longo do documento. Decidimos utilizar este filtro, pois assim possibilitava-nos também contar o número de *records* que cada extrato tinha.

```
1 BEGIN { RS="\n"; print "NºExtract -> NºRecords"; }
2
3 $1 == 1 {extracts++;}
4 $1 != "" {lines[extracts]++;}
5
6
7 END { for(i in lines) {print i " -> " lines[i] }; print "\nTotal Extracts: " extracts;}
```

Figura 1: Especificação GAWK

```
176 -> 63
177 -> 46
178 -> 36
179 -> 111
180 -> 41
181 -> 31
182 -> 20
183 -> 35
184 -> 281
185 -> 34
186 -> 85
187 -> 31
188 -> 66
189 -> 71
190 -> 7

Total Extracts: 190
```

Figura 2: Exemplo de output no ficheiro fl0

3.2 Ex B - Personagens do Harry Potter

Os ficheiros *harrypotter* fornecidos para este trabalho, com organização semelhante aos anteriores, contêm extratos de um livro do *Harry Potter*. Com isto foi requerido que se retirassem todos os nomes das personagens do *Harry Potter* que aparecem neste extratos. Para isto, com base no formato, concluímos que a coluna 4 e 5 contém a informação do tipo de palavra é o registo referente. Como nós procurávamos todos os **Nomes Próprios** existentes no livro, guardamos todas as palavras que contém **NP** na coluna referida. Para facilitar a leitura do resultado, construímos uma página

```
BEGIN { RS="\n"; flag_title = 0; i = 0

$3 == "titulo", $3=="*" {
    if($3 == ":"){
        flag_title = 0;
    }
    if(flag_title == 1){
        title[i++] = $2;
    }
    if($3 == "harry_potter"){
        flag_title = 1;
        title[i++] = $2;
    }
}

# $2 is the word in it's "true form"
$5 == "NP" {words[$3]++;}

END { generateHTML();}
```

Figura 3: Especificação GAWK para ex. B

HTML com uma tabela contendo todos os nomes. Possui também um título referente ao título do livro a que se refere o ficheiro tratado. Isto foi realizado, usando a habilidade do GAWK para realizar ações dentro de dois padrões (uma espécie de *BEGIN* e *END*). Como o título aparece sempre nas linhas entre **titulo** e no máximo um *, definimos estes como extremos dos padrões e, dentro, verificando quando é que começa o título (usando uma *flag*) e acaba (quando chega a um :).

Infelizmente, para além de nomes de personagens, encontra também outras palavras. Não foi encontrada solução para isto senão o uso de um di-


```

function generateHTML() {
    print "<html>\n<meta charset='UTF-8'>\n<header>\n<h1>Livro de Harry Potter</h1>\n</header>"> "harry_character.html"
    print " <body><h3>"> "harry_character.html"
    for(i in title) {
        print title[i], " "> "harry_character.html";
    }
    print "</h3><table border='1' style='float: left'>"> "harry_character.html"
    print "<tr><th> Personagens </th></tr>"> "harry_character.html"
    for (i in words) {
        print "<tr><td>", i , "</td></tr>"> "harry_character.html";
    }
    print "</table></body></html>"> "harry_character.html"
}

```

Figura 4: Função que gera ficheiro HTML para ex. B

cionário, contendo todos os nomes das personagens e usando esta como referencia do que é ou não de facto o nome de uma personagem. Esta não foi executada pois achamos uma solução difícil de implementar pela falta deste mesmo dicionário. Em baixo apresentamos o resultado do programa.

Livro de Harry Potter	
Harry_Potter and the Chamber of Secrets_Autor	
Index	Personagens
0	bruxas velhas
1	ron a hermione
2	salão nobre
3	ron a dumbledore
4	imundicie
5	ano
6	x.
7	janotas
8	câmara dos segredos tradução de isabel fraga editorial presença ficha técnica
9	chefe de turma
10	era colin creevey
11	d.
12	urgh
13	fim-de
14	after-eight
15	expresso de hogwarts
16	polisuco
17	isabel fraga capa
18	hallowe'en
19	ernie mcmillan dos hufflepuff
20	peter weasley
21	porque
22	ernie
23	madam pomfrey
24	guia de gilderoy lockhart
25	semana
26	encanta

Figura 5: Exemplo do output do ficheiro harrypotter2

3.3 Ex C - Listar Substantivos, Verbos, Adjectivos, Advérbios num ficheiro *HTML*

Neste exercício é proposto calcular um conjunto de listas que contenham, em cada uma delas, os verbos, substantivos, adjectivos e advérbios contidos num ficheiro.

Numa primeira análise do que era pedido foi verificado que existe um campo nos ficheiros denominado de *pos*, a classe morfológica da palavra a que se refere. Detetado este campo facilmente se criaram quatro *arrays* unidimensionais em que cada um continha o conjunto de palavras de cada tipo. De seguida é verificado linha a linha qual a primeira letra na posição *pos*:

- V - se verbo;
- N - se substantivo;
- A - se Adjectivo;
- R - se Advérbio.

```
BEGIN { RS="\n"; }  
  
index($4, "V") == 1 {verbs[$2]++;}  
index($4, "N") == 1 {nouns[$2]++;}  
index($4, "A") == 1 {adjectives[$2]++;}  
index($4, "R") == 1 {adverbs[$2]++;}
```

Figura 6: Especificação *GAWK*

Para a criação do ficheiro *HTML* foi criada uma função, chamada no fim do *script* que percorre os arrays um a um preenchendo os espaços de código que depois de completos são redireccionados para o ficheiro *.html* a ser lido pelo *browser*. Na figura seguinte podemos ver essa função.

```

function generateHTML() {
    print "<html>\n<meta charset=\"UTF-8\">\n<header>\n<h1>Dicionario Implícito</h1>\n</header>"> "word_types.html"

    print "<body><table border='1' style='float: left'>"> "word_types.html"
    print "<tr><th> Verbo </th></tr>"> "word_types.html"
    for (i in verbs) {
        | print "<tr><td>", i , "</td></tr>"> "word_types.html";
    }
    print "</table>" > "word_types.html"

    print "<table border='1' style='float: left'>"> "word_types.html"
    print "<tr><th> Substantivo </th></tr>"> "word_types.html"
    for (j in nouns) {
        | print "<tr><td>", j , "</td></tr>" > "word_types.html";
    }
    print "</table>" > "word_types.html"

    print "<table border='1' style='float: left'>"> "word_types.html"
    print "<tr><th> Adjectivo </th></tr>"> "word_types.html"
    for (k in adjectives) {
        | print "<tr><td>", k , "</td></tr>" > "word_types.html";
    }
    print "</table>" > "word_types.html"

    print "<table border='1' style='float: left'>"> "word_types.html"
    print "<tr><th> Advérbio </th></tr>"> "word_types.html"
    for (w in adverbs) {
        | print "<tr><td>", w , "</td></tr>" > "word_types.html";
    }
    print "</table>" > "word_types.html"
}

```

Figura 7: Função que gera ficheiro HTML

O resultado final da execução deste programa é mostrado na figura seguinte:

Lista de Palavras

Verbo	Substantivo	Adjectivo	Advérbio
poder	corrupção	envolventes	ideologicamente
desenhou	poder	européia	de acordo
tenho	prémio	diferente	Desde então
está	direita	bom	ilegalmente
tendo	autorizações	humanitárias	Mais
terminariam	cooptação	económica	também
dirigidas	truque	musical	mal
faz	Antye Greie	incidente	cá
contribuir	memória	draconianas	Antes
rondar	digressão	anterior	designadamente
cultivada	portugueses	pequenas	sá
tido	Administração Interna Eduardo Cabrita	consciente	lá
caminha	exibição	adjunto	antes
implicar	reaparecimento	essenciais	muito
prolongaram	Mário	produtivas	eventualmente
consideradas	ano	maior	mais
fosse	tipo	reveladoras	isto aí
confirmados	Vera	cultural	hoje
acabou	Instituto Nacional de Estatística	eventual	docilmente
contam	Jesus	rico	sá

Figura 8: Exemplo de output no ficheiro fl1

3.4 Ex D - Determinar o dicionário implícito no corpora

Este exercício consiste em criar o dicionário implícito no corpora dos ficheiros introduzidos como argumento do programa. O dicionário implícito é composto por lema, palavras dele derivadas e pos.

Para resolver este exercício foi criado um *array* multidimensional para armazenar os lema, pos e palavras derivadas encontradas. Em que o lema ocupa a primeira componente do *array*, as palavras derivadas ocupam o *array* lá contido e a pos ocupa a posição combinada. Antes de serem inseridas, a cada componente na estrutura de dados é feita uma verificação através de expressões regulares se o lema que estamos a inserir apenas contém letras, *hífens* e *underscores*. Podemos ver as especificação do programa e o seu resultado nas imagens seguintes:

