

Processamento de Linguagens

Carlos Pedrosa a77320

David Sousa a78938

Manuel Sousa a78869

24 de Março de 2018

Resumo

Este projeto tem como principal objetivo a interação por parte dos alunos com ferramentas de apoio à programação. Neste sentido, permite assim, aumentar a capacidade destes relativamente à escrita de expressões regulares como motor para a filtração e transformação de textos. Em suma, este relatório pretende sumarizar todos os esforços efetuados para alcançar o objetivo proposto. Primeiramente, introduziremos o problema proposto, passaremos então pela conceção da solução e por último faremos uma análise crítica a todo o trabalho elaborado.

Conteúdo

1	Introdução	2
2	Descrição do problema	2
3	Concepção da solução	2
3.1	Alíneas propostas	2
3.2	Funcionalidades extras	5
3.3	Considerações importantes e resultados obtidos	6
4	Conclusões	8
5	Bibliografia	9
6	Anexos	10

1 Introdução

Este trabalho foi-nos proposto no âmbito da unidade curricular de Processamento de Linguagens e tem como principal objetivo fazer com que haja, por parte dos alunos, um maior contacto com novas ferramentas de programação.

Nesse sentido, foi-nos apresentado um trabalho que consistia principalmente na identificação de padrões em texto-fonte através de expressões regulares. Efetivamente, devido à política de seleção implementada pelo docente, o nosso grupo irá se debruçar pelo primeiro enunciado apresentando. Para a elaboração, utilizamos a ferramenta indicada na resolução de questões relativas a processamento de texto, o *Gawk*.

O *Gawk* é uma linguagem que permite processar textos segundo diversos padrões. Uma característica importante é que ficheiros de entrada são processados linha a linha. Caso nenhum padrão seja especificado todas as linhas são processadas. Assim, as linhas são separadas em campos, que são divididos segundo a premissa **FS** (field separator). Outra informação relevante é que os ficheiros de entrada não são afetados pelo processamento.

Em suma, pretendemos nas secções a seguir demonstrar o esforço envolvido por parte de todos os elementos do grupo na construção de soluções para o guião enunciado.

2 Descrição do problema

O enunciado começa por definir o que são corpora. Neste sentido, é apresentado um formato que tem por base este tipo de documentos. Este ficheiro tem como nome CETEMPúblico e um aspeto interessante é que usa tags xml para a anotação frásica, e colunas separadas por tab para a informação morfossintática de cada palavra. Neste sentido, são apresentadas premissas que se prendem com o processamento de CETEMPúblico, usando para isso as ferramentas de apoio à programação. Assim, o desafio prende-se sobretudo com resolução destas questões extraíndo todas as informações necessárias.

3 Concepção da solução

Ao longo do trabalho fomos testando os métodos por nós elaborados através do uso de diferentes testes. Nos seguintes pontos exemplificaremos os passos dados para a resposta às diferentes interrogações fazendo, sempre que necessário, alusão aos ficheiros desenvolvidos.

3.1 Alíneas propostas

- Contar o número de extratos, parágrafos e frases

De modo a contabilizar o número de extratos começamos por analisar os ficheiros fornecidos para que, desta forma, fosse possível encontrar padrões que facilitassem a resolução da alínea pretendida. Cada extrato encontrava-se delimitado pela tag `<ext ...>` e `</ext>`. Assim, apenas contabilizamos o número de vezes que a segunda tag surgiu ao longo do documento. Efetivamente, consideramos mais prático ter em conta o fecho da tag uma vez que esta surge de modo singular. Para as restantes contagens usamos o mesmo processo. O uso de expressões regulares adequadas facilitou bastante o processo.

```
BEGIN {  
$1 ~ /</ext>/ { e++ }  
$1 ~ /</p>/ { p++ }  
$1 ~ /</s>/ { f++ }  
END {  
    print "Número de extratos: " e "\n" "Número de parágrafos: " p "\n" "Número de frases: " f  
}
```

Figura 1: Ficheiro produzido para a 1ª alínea.

- Extrair a lista das multi-word-expressions e respetivo número de ocorrências

Para esta alínea foi importante a definição do campo **FS**. Constatamos que cada fator no ficheiro se encontrava separado por *tab*. Nesse sentido foi este o valor que **FS** tomou. De salientar a declaração de uma *string* auxiliar, útil na concatenação das expressões. Uma vez mais fizemos uso das tag que o documento fornecido apresentava. Sempre que a tag *<mwe ...>* aparecia, inicializamos o contador *C* com o valor um e utilizamos a função *next* facultada pelo *Gawk*. Isto deve-se ao facto de o texto que necessitávamos de retirar não se encontrar nessa linha mas sim na seguinte. Caso o contador apresentasse o valor 1 e caso ainda nos encontrássemos dentro da tag *<mwe ...>* (esta ainda não fechou), recolhíamos o \$1 (primeiro campo que resulta da divisão segundo o FS), incrementávamos o contador e passamos para a linha seguinte. De referir que o contador serve apenas de controlo para o espaço produzido entre as palavras no output. Efetivamente, este não deveria aparecer no início de cada linha que será imprimida. Por fim, quando a tag final desta secção é atingida o contador toma o valor de zero e guardamos a string no array associativo para que seja possível calcular o número de ocorrências. O *END* apenas imprime para um ficheiro, que achamos por bem criar para facilitar a visualização dos resultados obtidos.

\$1	\$2	\$3	\$4	\$5	\$6	\$7	\$8	\$9	\$10	\$11	\$12
Ontem	pol	92a	ontem	ADV	0	0	0	ADVL>	0	0	0

Figura 2: Exemplo da divisão obtida através do FS.

```
BEGIN { FS = "\t"; S = "" }

$1 ~ /<mwe.*/ { c = 1; next; }
c == 1 && ($1 !~ /<\mwe>/) { S = $1; c++; next; }
c > 1 && ($1 !~ /<\mwe>/) { S = S " " $1; }
$1 ~ /<\mwe>/ { c = 0; dic[S]++; S = ""; }

END {
    for (i in dic) {
        print "Expressão: " i, dic[i] > "test.txt";
        sum += dic[i];
    }
    print sum;
}
```

Figura 3: Ficheiro produzido para a 2ª alínea.

- Calcule a lista dos verbos PT: (Lema, para palavras com pos=V) e respetivo número de ocorrências

Mais uma vez definimos o **FS** como *"|t"* por uma questão de conveniência. Segundo o exemplo, o campo \$5 é onde se apresenta o *part of speech* referido no enunciado (por exemplo o *V*). Novamente, decidimos usar um array associativo, uma vez que este tipo de estrutura permite guardar e contar o número de ocorrências dos resultados mais facilmente. Por fim, apenas imprimimos os valores obtidos.

```

BEGIN { FS = "\t" }

$5 == "V" { A[$4]++ }

END {
    for (i in A) {
        print i;
        print A[i];
    }
}

```

Figura 4: Ficheiro produzido para a 3ª alínea.

- Determinar o dicionário implícito no corpora – calcule a lista das palavras associando-lhes os possíveis(lema, pos)

Na resolução desta alínea é importante destacar o uso de arrays associativos multi-nível. Novamente aproveitamos as tags existentes nos ficheiros fornecidos.

```

BEGIN { FS = "\t" }

$1 !~ /^</ { A[$1][$4][$5]++; }

END {
    for (i in A) {
        for (j in A[i]) {
            for (k in A[i][j]) {
                print "i-> " i, " j-> " j, " k-> " k, " | Valor-> " A[i][j][k]
            }
        }
    }
}

```

Figura 5: Ficheiro produzido para a 4ª alínea.

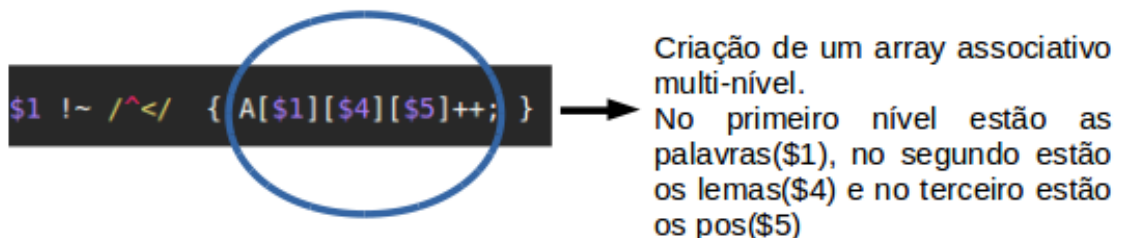


Figura 6: Exemplo para a 4ª alínea.

3.2 Funcionalidades extras

Uma vez que era encorajado o esforço extra decidimos construir ficheiros *Gawk* para dar resposta a alíneas que não eram abrangidas pelo enunciado. As nossas implementações basearam-se na construção de HTML que permitiam uma melhor organização dos dados recolhidos.

- Restabelecimento do texto contido nos documentos

Consideramos que seria útil construir um texto ‘corrido’ com os documentos que nos eram fornecidos. De facto, a forma como texto é apresentado não é, de todo, a mais benéfica para uma leitura eficiente. Assim, recorrendo ao HTML, construimos um índice com o número de extrato correspondente, em que em cada hiperligação se encontra o texto pretendido. É importante realçar o uso das funções *split* e *gsub* neste ficheiro. A primeira foi útil para dividir por espaços toda a tag recolhida. Efetivamente, foi necessário usar este processo para obter o número de cada extrato. Neste sentido, para obtermos o número de extrato que estava a ser tratado apenas teríamos de aceder ao segundo campo do array utilizado no *split*. A função *gsub*, por sua vez, foi utilizada para eliminar o ‘n=’ que se encontrava neste (ver figura 7).

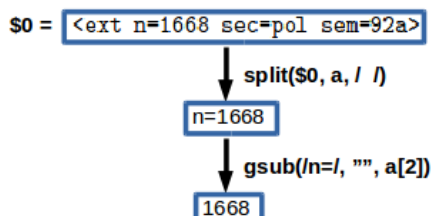


Figura 7: Exemplificação da utilização de *split* e *gsub*.

Lista de Extratos

- [Extrato 1665](#)
- [Extrato 1666](#)
- [Extrato 1667](#)
- [Extrato 1668](#)
- [Extrato 1669](#)
- [Extrato 1670](#)
- [Extrato 1671](#)
- [Extrato 1672](#)
- [Extrato 1673](#)
- [Extrato 1674](#)

Figura 8: Novo ficheiro produzido para a 4ª alínea.

Extrato 1669

O abrandamento da economia nipónica intensificou-se no terceiro trimestre do ano em curso , ao registar uma queda de 1,6 por cento na produção , segundo anunciou a Agência de Planeamento Económico . Ao mesmo tempo , o produto nacional bruto recuou 0,4 por cento , quando comparado com os resultados dos três meses precedentes .

Desemprego melhora nos Estados Unidos ...

Figura 9: Exemplo do resultado obtido.

- Alteração da alínea 4 com a construção de uma tabela

Consideramos que a criação de um dicionário poderia ser feita de um modo mais prático. Deste modo, facilitamos a visualização dos termos contidos neste com a criação de uma tabela também em HTML.

```
BEGIN {
  FS = "\t"
  s = ""
  print "<html><head><style>\n" > "tabelaDic.html"
  print "table { border-collapse: collapse; width 100%; }\n" > "tabelaDic.html"
  print "th, td { text-align: left; padding: 8px; }\n" > "tabelaDic.html"
  print "tr:nth-child(even){ background-color: #f2f2f2 }\n" > "tabelaDic.html"
  print "th { background-color: #3A80F2; color: white; }\n" > "tabelaDic.html"
  print "</style><meta charset='UTF-8'> </head><body>\n" > "tabelaDic.html"
}
```

No exemplo acima apresentado, podemos ver o separador de campo como sendo o *tab*. Assim, criamos os ficheiros imprimindo as premissas HTML para estes usando redirecionamento.

Palavra	Radical	Part Of Speech
autarca	autarca	N
ónus	ónus	N
participara	participar	V
acusasse	acusar	V
Salisb	Salisb	PROP
aconteceram-me	acontecer+eu	V_mv_fmc_vN+PERS_refl
Desilusão	desilusão	N

Figura 10: Exemplo do resultado obtido.

3.3 Considerações importantes e resultados obtidos

No decorrer do trabalho não alteramos o campo **RS** uma vez que o valor *default* deste, enquadrava-se nos objetivos pretendidos. De facto, a escolha apropriada de um **FS** e um **RS** facilita bastante o projeto em causa. Por questões de simplificação do relatório, os resultados apresentados a seguir dizem respeito ao ficheiro *micro-Cetempublico01.txt*. É importante referir que para ficheiros maiores os resultados obtidos continuam dentro do pretendido(ver anexo). Apenas realçar que o último documento, *CETEMPublicoAnotado2018-p1*, era bastante extenso, pelo que não incluímos, neste relatório, os resultados que obtivemos com este. No entanto, apesar de output levar mais tempo a ser produzido, conseguimos obter respostas satisfatórias para as questões fornecidas.

```
david@David Perguntal $ gawk -f extractNumber.gawk ../micro-Cetempublico01.txt
Número de extratos: 1295
Número de parágrafos: 3029
Número de frases: 6701
```

Figura 11: Resultados obtidos na alínea 1.

```

Expressão: em trânsito 1
Expressão: a qual 13
Expressão: ao contrário de 2
Expressão: à margem 1
Expressão: dívida pública 3
Expressão: em cima 4
Expressão: até mesmo 1
Expressão: construção civil 2
Expressão: com respeito 1
Expressão: estado de emergência 1
Expressão: valor patrimonial 1
Expressão: À margem 1
Expressão: fim de semana 4
Expressão: a partir de 15

```

Figura 12: Resultados obtidos na alínea 2.

```

retomar
9
despedir
1
vir
125
homenagear
1
recém-licenciar
1
permitir
50
ver
155

```

Figura 13: Resultados obtidos na alínea 3.

```

i-> licenciatura | j-> licenciatura | k-> N | Valor-> 1
i-> porque | j-> porque | k-> KS | Valor-> 112
i-> porque | j-> porque | k-> ADV | Valor-> 5
i-> Mondim | j-> De=Mondim | k-> PROP | Valor-> 1
i-> regressa | j-> regressar | k-> V | Valor-> 1
i-> rubrica | j-> rubrica | k-> N | Valor-> 1
i-> vigária | j-> vigária | k-> N | Valor-> 1

```

Figura 14: Resultados obtidos na alínea 4.

4 Conclusões

De um modo geral, podemos concluir que os objetivos foram concluídos. De facto, foi possível responder a todas as questões pedidas pelo docente bem como acrescentar outras que consideramos serem úteis. Assim, é importante realçar o interesse revelado por todos os elementos no decorrer da resolução do trabalho.

Os desafios da resolução deste projeto prenderam-se sobretudo com a escolha apropriada do FS. Para além disso, usamos funções(fornecidas pelo Gawk) com as quais não possuíamos contacto prático. No entanto, depois de uma breve pesquisa o restante trabalho surgiu naturalmente.

Em jeito de conclusão, o presente trabalho serviu para um aprimoramento das nossas aptidões relativamente a uma ferramenta de processamento de texto, Gawk. Efetivamente, foi possível ao grupo concluir o quão poderosas podem ser estas ferramentas. Assim, este trabalho formatou a nossa maneira de pensar aquando da resolução de um problema, tornando-nos mais objetivos e racionais.

5 Bibliografia

- <http://www.programacaoprogessiva.net/2012/07/shell-awk-part-i-o-que-e-e-para-que.html> acessado no dia 21 pelas 14h;
- <http://natura.di.uminho.pt/~jj/pl-18/> acessado no dia 13 pelas 15h35m.

6 Anexos

Código fonte dos exercícios extra:

```
BEGIN
{
FS = "\t";
f = "<li><a href = '%s'> %s </a> </li> \n";
f2 = "<html><body><p> %s ";
print "<h1 style='color:blue;'> <i> Lista de Extratos </i> </h1> \n" > indicegeral.html";
print "<html><head><title>Lista de Extratos</title><meta charset='UTF-8'> </head> \n" > "indicegeral.html";
}

$1 ~ /<ext.* /
{
split($0, a, / /);
gsub(/n=/, "", a[2]);
printf (f,"Extrato " a[2] ".html","Extrato " a[2]) > "indicegeral.html";
printf "<html><head><meta charset='UTF-8'> </head> \n" > "Extrato " a[2] ".html";
printf ("<h1> <i> <b style = 'color:red'> %s </b> </i> </h1> \n", "Extrato " a[2]) > "Extrato " a[2] ".html";
print "<html><body>" > "Extrato " a[2] ".html"
}

$1 ~ /<p.* /      { print "<p><t>" > "Extrato " a[2] ".html"}

$1 !~ /<.* /      { printf ($1 " ") > "Extrato " a[2] ".html" }

$1 ~ /<\/p> /      { print "</p>" > "Extrato " a[2] ".html" }

$1 ~ /<\/ext> /    { print ("</body></html>") > "Extrato " a[2] ".html" }

END {}

BEGIN
{
FS = "\t"
s = ""
print "<html><head><style>\n" > "tabelaDic.html"
print "table { border-collapse: collapse; width 100%; }\n" > "tabelaDic.html"
print "th, td { text-align: left; padding: 8px; }\n" > "tabelaDic.html"
print "tr:nth-child(even){ background-color: #f2f2f2 }\n" > "tabelaDic.html"
print "th { background-color: #3A80F2; color: white; }\n" > "tabelaDic.html"
print "</style><meta charset='UTF-8'> </head><body>\n" > "tabelaDic.html"
}

$1 !~ /^</ { A[$1][$4][$5]++; }

END {
print "<table><tr> <th>Palavra</th> <th>Radical</th> <th>Part Of Speech</th> </tr>" > "tabelaDic.html"

for (i in A) {
for (j in A[i]) {
for (k in A[i][j]) {
if (N >= 1) {
s = s ", " k
} else {
s = k
}
}
}
}
```

```

        N++
    }
    print "<tr><td>" i "</td> <td>" j "</td> <td>" s "</td></tr>" > "tabelaDic.html"
    s = ""
    N = 0
    }
}

print "</table></body></html>" > "tabelaDic.html"
}

```

Resultados obtidos com a utilização do ficheiro *Cetempublico01.txt*:

```

david@David Pergunta1 $ gawk -f extractNumber.gawk ../Cetempublico01.txt
Número de extratos: 5454
Número de parágrafos: 12843
Número de frases: 28863

```

Figura 15: Resultados obtidos na alínea 1 com outro documento.

```

Expressão: inimigo figadal 1
Expressão: salários mínimos 2
Expressão: próximo de 8
Expressão: em boa medida 1
Expressão: de orelha 1
Expressão: De agora 1
Expressão: no geral 1
Expressão: a todo o pano 1
Expressão: À luz 1
Expressão: em trânsito 2
Expressão: ao contrário de 6
Expressão: a qual 33

```

Figura 16: Resultados obtidos na alínea 2 com outro documento.

```

achar
28
indexar
1
acatar
3
retaliar
1
contra-atacar
4

```

Figura 17: Resultados obtidos na alínea 3 com outro documento.

```
i-> verdade | j-> verdade | k-> IN | Valor-> 3
i-> ficção | j-> ficção | k-> N | Valor-> 1
i-> admiravelmente | j-> admiravelmente | k-> ADV | Valor-> 2
i-> Barnard | j-> John=Barnard | k-> PROP | Valor-> 1
```

Figura 18: Resultados obtidos na alínea 4 com outro documento.

Lista de Extratos

- [Extrato 1](#)
- [Extrato 2](#)
- [Extrato 3](#)
- [Extrato 4](#)
- [Extrato 5](#)
- [Extrato 6](#)
- [Extrato 7](#)
- [Extrato 8](#)
- [Extrato 9](#)
- [Extrato 10](#)
- [Extrato 11](#)

Figura 19: Resultados obtidos na alínea extra 1 com outro documento.

Palavra	Radical	Part Of Speech
invólucros	invólucro	N
envolvente	envolvente	ADJ
resolve	resolver	V
mutilados	Recentes=desejos=mutilados	PROP
mutilados	mutilado	ADJ

Figura 20: Resultados obtidos na alínea extra 2 com outro documento.