Universidade do Minho
Mestrado Integrado em Engenharia Informática
Departamento de Informática

Processamento de Linguagens

Trabalho Prático 1 - Exercício 2

António Sérgio Alves Costa A78296
Isabel Sofia da Costa Pereira A76550
Maria de La Salete Dias Teixeira A75281

25 de Março de 2018

Conteúdo

1 Introdução

No âmbito da unidade curricular de Processamento de Linguagens, foi-nos proposto o desenvolvimento de programas *awk* que processem ficheiros de textos preanotados com Freeling.

Estes ficheiros armazenam uma vasta quantidade de textos (extratos) que, por sua vez, entre as várias informações neles contidas, dispõem também de indicações sobre a gramática. Além disso, o formato Freeling separa as diferentes colunas com espaços e os vários extratos com uma linha em branco.

Através da linguagem *awk* é possível analisar o conteúdo das colunas de um texto, linha a linha. Dessa forma, é viável utilizar esta linguagem para o processamento de textos e efetuar a manipulação dos mesmos.

2 Descrição do Trabalho e Ánalise dos Resultados

Neste trabalho foram propostos quatro desafios. Para cada um destes, é descrito o processo de resolução do problema, explicitando a respetiva implementação em *awk* e os resultados obtidos.

Além dos desafios sugeridos foram desenvolvidos mais cinco programas que respondem a novas questões que o grupo considerou pertinentes.

2.1 Desafio 1: contar o número de extratos

O objetivo deste desafio baseia-se em determinar o número de extratos que cada documento disponibilizado possui.

Sabendo que cada extrato é separado por uma linha em branco, é essencial desenvolver um programa que incremente uma váriavel sempre que encontre uma linha cujo número de campos/colunas seja igual a zero.

Desta forma, o programa em *awk* desenvolvido foi o seguinte:

```
BEGIN {FS=" "}
NF==0 {conta++}
END {print conta}
```

Figura 1: Programa em *awk* do desafio 1

Após a implementação do progama é possível obter resultados tais como:

```
sofia@sofia-X750JN ~/Desktop/3 ANO/PL/projeto $ gawk -f extratos.awk fl0
190
sofia@sofia-X750JN ~/Desktop/3 ANO/PL/projeto $ gawk -f extratos.awk fl1
190
sofia@sofia-X750JN ~/Desktop/3 ANO/PL/projeto $ gawk -f extratos.awk fl2
102
sofia@sofia-X750JN ~/Desktop/3 ANO/PL/projeto $ gawk -f extratos.awk harrypotter1
5569
sofia@sofia-X750JN ~/Desktop/3 ANO/PL/projeto $ gawk -f extratos.awk harrypotter2
5432
sofia@sofia-X750JN ~/Desktop/3 ANO/PL/projeto $ gawk -f extratos.awk fl0 fl1 fl2 harrypotter1 harrypotter2
11483
```

Figura 2: Comandos para execução do programa do desafio 1 e respetivas respostas

2.2 Desafio 2: calcular a lista das personagens do Harry Potter (nomes próprios) e respectivo número de ocorrências

Com este desafio pretende-se encontrar o nome próprio de todas as personagens que surgem nos documentos *harrypotter1* e *harrypotter2* e calcular o número de vezes que cada uma destas ocorre nos textos.

Para a realização deste desafio foi necessário estudar o conteúdo de cada coluna dos ficheiros *harrypotter*. Como o objetivo é filtrar nomes próprios, foi necessário utilizar a coluna número cinco que informa a qual classe gramatical é que a palavra em questão pertence. Desta forma, foi essencial efetuar a filtragem das linhas através das colunas cinco que contivessem o valor NP (nome próprio).

Além disso, para armazenar o nome encontrado e o número de vezes que este aparece, é conveniente a existência de um array. Esse array contém então como posição o nome da personagem, que se encontra na coluna número dois, e como conteúdo o número de vezes que essa personagem surgiu no documento.

Por fim, é escrito num ficheiro com o nome *personagensHP.txt*, linha a linha, o número de vezes que uma certa personagem aparece e o nome da personagem correspondente. Neste ponto, achou-se proveitoso o uso de um *pipe sort* para organizar por ordem decrescente os números de ocorrências encontrados. Desta forma, as primeiras entradas de *personagensHP.txt* especificam as personagens que são mencionadas mais vezes ao longo do documento. É também importante salientar que, devido à composição do ficheiro, não foi possível para o grupo separar as personangens de outros nomes próprios como, por exemplo, nomes de cidades ou edifícios, sendo que o ficheiro resultante contém todos os nomes próprios presentes nos ficheiros originais.

Assim, o programa em *awk* desenvolvido foi o seguinte:

```
BEGIN {FS=" "}
$5 ~ /NP/ {array[$2]++}
END {for (i in array) print array[i], i | "sort -n -r > personagensHP.txt"}
```

Figura 3: Programa em *awk* do desafio 2

A elaboração deste progama permite obter resultados como os apresentados de seguida.

```
sofia@sofia-X750JN ~/Desktop/PL $ gawk -f HPnomes.awk harrypotter1 harrypotter2
```

Figura 4: Comando para execução do programa do desafio 2

```
2571 Harry
1064 Ron
537 Hermione
484 Hagrid
280 Dumbledore
244 Malfoy
242 Snape
191 Lockhart
182 Mc_Gonagall
177 Gryffindor
158 Slytherin
153 Dudley
145 Vernon
144 Dobby
142 Hogwarts
129 Neville
125 Fred
116 Harry_Potter
```

Figura 5: Respostas ao desafio 2 - primeiras entradas de *personagensHP.txt*

2.3 Desafio 3: calcular a lista dos verbos, substantivos, adjectivos e adverbios PT e criar um ficheiro com cada uma destas listas

O desafio número três propõe a elaboração de quatro ficheiros de texto. Cada um desses ficheiros terá de armazenar, respetivamente, os verbos, substantivos, adjectivos e adverbios presentes no documento pretendido.

Para tal, é crucial analisar a composição dos documentos disponibilizados. Como alguns dos documentos apresentam uma estrutura ligeiramente diferente, foi necessário encontrar uma coluna com número igual para todos que permitisse identificar verbos, substantivos, adjectivos e adverbios. Determinou-se que a coluna que apresenta essa característica é a quatro. Nesta coluna, os verbos são identificados no início pela letra V, os substantivos pela letra N, os adjetivos pela letra A e os adverbios pela letra R.

Tendo essa informação em conta, para todas as linhas que não estejam em branco, criou-se uma expressão condicional que, consoante a letra encontrada na coluna quatro, guarda no array verbos, substantivos, adjectivos ou adverbios a palavra encontrada como índice deste e como conteúdo o número de vezes que essa palavra surge.

Por fim, quando os documentos estão todos processados e os verbos, substantivos, adjectivos e adverbios todos encontrados, é criado um ficheiro cor-

respondente para cada um e escrita neste a informação encontrada (número de ocorrências e palavra) de forma ordenada descrescentemente pelas ocorrências da palavra.

Tendo em conta o que foi mencionado acima, o programa em *awk* implementado segue o seguinte formato:

```
BEGIN {FS=" "}

NF>0 {if ($4 ~ /^V/) verbos[$2]++;
      if ($4 ~ /^N/) substantivos[$2]++;
      if ($4 ~ /^A/) adjetivos[$2]++;
      if ($4 ~ /^R/) adverbios[$2]++ }

END {for(i in verbos) print verbos[i], i | "sort -n -r > verbos.txt";
     for(i in substantivos) print substantivos[i], i | "sort -n -r > substantivos.txt";
     for(i in adjetivos) print adjetivos[i], i | "sort -n -r > adjetivos.txt";
     for(i in adverbios) print adverbios[i], i | "sort -n -r > adverbios.txt"}
```

Figura 6: Programa em *awk* do desafio 3

Após a execução do programa *awk*, é possível examinar o ficheiro *verbos.txt*, *substantivos.txt*, *adjectivos.txt* e *adverbios.txt*.

```
sofia@sofia-X750JN ~/Desktop/PL $ gawk -f listas.awk harrypotter1 harrypotter2 fl0 fl1 fl2
```

Figura 7: Comando para execução do programa do desafio 3

1479 disse	2571 Harry	243 grande	1952 não
750 é	1064 Ron	194 melhor	761 mais
627 estava	537 Hermione	118 enorme	626 Não
527 tinha	484 Hagrid	91 maior	450 quando
520 era	304 coisa	83 claro	351 muito
450 ter	299 olhos	79 cheio	297 lá
420 foi	281 cabeça	71 primeira	287 enquanto
343 ser	280 Dumbledore	71 alto	280 bem
337 fazer	265 porta	63 grandes	255 aqui
311 perguntou	244 Malfoy	63 baixo	254 tão
266 está	242 Snape	61 possível	242 já
259 ver	226 voz	58 preciso	234 ainda
232 dizer	212 professora	57 cheia	231 é_que
216 É	212 ar	56 pequeno	214 onde
211 tinham	191 Lockhart	55 seguinte	211 também

Figura 8: Respostas ao desafio 3 - os quatro ficheiros gerados: verbos, substantivos, adjetivos e adverbios

2.4 Desafio 4: determinar o dicionário implícito no corpora – lista contendo o lema, pos e palavras dele derivadas

Neste desafio é pedida a constução de um dicionário constituído pelo lema, pos e palavras derivadas presentes nos ficheiros. Tanto pela análise dos ficheiros como pelo enunciado do trabalho, confirmamos que o lema corresponde à coluna 3, o pos à coluna 5 e a palavra derivada à coluna 2. Para uma melhor leitura do dicionário e aspeto mais apelativo, decidiu-se criar um ficheiro *html* constituído por uma tabela.

Começou-se então por criar um documento com a syntax inicial de um ficheiro *html*. De seguida, estabeleceu-se uma entrada na tabela com o conteúdo da coluna 3 (lema), 5 (pos) e 2 (palavra) por cada linha lida que contenha apenas letras na coluna três. Este requisito foi garantido pelo uso da expressão regular ∧[A-Za-z]+$. Teve-se também em atenção a inserção das linhas no dicionário por ordem alfabética através do comando *sort* e a não adição de conteúdo repetido através do comando *uniq*, ambos aplicados por um pipe, ou seja, só se insere entradas únicas no dicionário para não haver redundância de informação.

Por fim, coloca-se a syntax final num ficheiro com extensão *html*. Para isso, é necessário direcionar para um ficheiro *html* no terminal.

O programa *awk* resultou no seguinte:

```
BEGIN {FS=" ";
       print "<htlm>\n\t<body>\n\t<table border='1'><thead><tr><th>lema</th><th>pos</th><th>palavra derivada</th></tr></thead><tbody>"}

NF>0 && $3 ~ /^[A-Za-z]+$/ {print "<tr><td>" $3 "</td><td>" $5 "</td><td>" $2 "</td></tr>" | "sort | uniq"}

END {print "</tbody></table>\n</body>\n</html>"}
```

Figura 9: Programa em *awk* do desafio 4

Após a execução do programa *awk*, é possível averiguar o conteúdo do ficheiro *dicionario.html*.

```
sofia@sofia-X750JN ~/Desktop/PL $ gawk -f dicionario.awk harrypotter1 harrypotter2 fl0 fl1 fl2 > dicionario.html
```

Figura 10: Comando para execução do programa do desafio 4

lema	pos	palavra derivada
aaaaaaaahhhh	NP	AAAAAAAAHHHH
aaaaaaarch	NP	AAAAAAARCH
aaargh	NP	Aaargh
aaargh	NP	AAARGH
abafador	NC	abafador
abafar	-	abafados
abafar	VMG	abafando
abafar	VMI	abafou
abafar	VMN	abafar
abafar	VMP	abafadas
abafar	VMP	abafada
abafar	VMP	abafados
abafar	VMP	abafado
abafo	NC	abafos
abafo	NC	abafo
abaixo	RG	abaixo

Figura 11: Respostas ao desafio 4 - ficheiro html

2.5 Extra 1: elaboração de textos

Para o primeiro extra decidiu-se criar um *pdf* com as frases de um ficheiro todas juntas, ou seja, um texto corrido normal.

Primeiramente, teve-se de criar um ficheiro *tex* com toda a sua syntax e juntar as palavras dos documentos. Assim sendo, começou-se por criar uma função, *beginlatex()*, que cria toda a syntax inicial de um ficheiro *latex*.

Seguidamente, teve-se de ler palavra a palavra e colocá-las dentro dum ficheiro passado como parâmetro. Este passo foi bastante simples já que todas as palavras se encontram na segunda coluna.

Apesar da utilização de alguns pacotes para lidar com caracteres especiais em *latex*, não se conseguiu resolver todos problemas, por isso decidiu-se trocar estes caracteres por outros facilmente equivalentes.

No final, foi necessário terminar a syntax *latex* e correr um comando para tornar o ficheiro *tex* num *pdf*. Para tal, utilizou-se a função *system* que permite executar comandos a partir dum ficheiro *awk*.

Desta forma, o programa *awk* resultou no seguinte:

```
BEGIN {
    FS=" ";
    ORS=" ";
    filename=ARGV[2];
    beginlatex();
    IGNORECASE=1;
}

$2 ~ /[õ,ó]/ {gsub(/[õ,ó]/,"o")}
$2 ~ /[ã,à,á]/ {gsub(/[ã,à,á]/,"a")}
$2 ~ /ç/ {gsub(/ç/,"c")}
$2 ~ /é/ {gsub(/é/,"e")}
$2 ~ /í/ {gsub(/í/,"i")}
$2 ~ /[_,{,}]/ {gsub(/[_,{,}]/," ")}
NF>0 {print $2 >> filename}
NF==0 {print "\n \\par" >> filename}

END{
    print "\n\\end{document}" >> filename;close(filename);
    str = "texi2pdf " filename;
    system(str);
}

function beginlatex(){
    print "\\documentclass{article}\n\\usepackage[utf8]{inputenc}\n\\usepackage[portuguese,brazil,english]{babel}\n\\title{PL}\n\\au
}
```

Figura 12: Programa em *awk* do extra 1

Posteriormente à execução do programa *awk* é praticável analisar o conteúdo do ficheiro gerado.

```
------------------------------------------------------------
~/asac/Universidade/PL/PL(asac) » awk -f textoFluido2.awk harrypotter1 hp1.tex
This is pdfTeX, Version 3.14159265-2.6-1.40.18 (TeX Live 2017/Arch Linux) (prelo
aded format=pdflatex)
 restricted \write18 enabled.
entering extended mode
LaTeX2e <2017-04-15>
Babel <3.18> and hyphenation patterns for 84 language(s) loaded.
(./hp1.tex (/usr/share/texmf-dist/tex/latex/base/article.cls
Document Class: article 2014/09/29 v1.4h Standard LaTeX document class
(/usr/share/texmf-dist/tex/latex/base/size10.clo))
(/usr/share/texmf-dist/tex/latex/base/inputenc.sty
(/usr/share/texmf-dist/tex/latex/base/utf8.def
(/usr/share/texmf-dist/tex/latex/base/t1enc.dfu)
(/usr/share/texmf-dist/tex/latex/base/ot1enc.dfu)
(/usr/share/texmf-dist/tex/latex/base/omsenc.dfu)))
(/usr/share/texmf-dist/tex/generic/babel/babel.sty
(/usr/share/texmf-dist/tex/generic/babel/switch.def)
(/usr/share/texmf-dist/tex/generic/babel-portuges/portuges.ldf
(/usr/share/texmf-dist/tex/generic/babel/babel.def
(/usr/share/texmf-dist/tex/generic/babel/txtbabel.def)))
(/usr/share/texmf-dist/tex/generic/babel-portuges/portuges.ldf)
(/usr/share/texmf-dist/tex/generic/babel-english/english.ldf))
No file hp1.aux.
[1{/var/lib/texmf/fonts/map/pdftex/updmap/pdftex.map}] [2] [3] [4]
Overfull \hbox (0.2784pt too wide) in paragraph at lines 98--99
[]\OT1/cmr/m/n/10 Quando o Dud-ley foi metido em a cama o en-cam-in-hou se para
 a sala chegando
[5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20]
[21] [22] [23] [24] [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] [35]
[36] [37] [38] [39] [40] [41] [42] [43] [44] [45] [46] [47] [48] [49] [50]
[51] [52] [53] [54] [55] [56] [57] [58] [59] [60] [61] [62] [63] [64] [65]
[66] [67] [68] [69] [70] [71] [72] [73] [74] [75] [76] [77] [78] [79]
Overfull \hbox (1.58385pt too wide) in paragraph at lines 2153--2154
[]\OT1/cmr/m/n/10 Quando Harry deu o primeiro passo em frente o a sala encheu s
e de murm^^Surios
[80] [81] [82] [83] [84] [85] [86] [87] [88] [89] [90] [91] [92] [93] [94]
[95] [96] [97] [98] [99] [100] [101] [102] [103] [104] [105] [106] [107]
[108] [109] [110] [111] [112] [113] [114] [115] [116] [117] [118] [119]
[120] [121] [122] [123] [124] [125] [126] [127] [128]
Overfull \hbox (1.05602pt too wide) in paragraph at lines 3438--3439
[]\OT1/cmr/m/n/10 - Parece en-tao que temos de de-sco-brir por nos proprios-dis
se o Ron o deixando
[129] [130] [131] [132] [133] [134] [135] [136] [137] [138] [139] [140]
Overfull \hbox (0.19511pt too wide) in paragraph at lines 3811--3812
[]\OT1/cmr/m/n/10 Se gan-has-sem o prox-imo jogo con-tra os Huf-flepuff ul-tra-
pas-sariam os Slytherin
[141] [142] [143] [144] [145] [146] [147] [148] [149] [150] [151] [152]
[153] [154] [155] [156] [157] [158] [159]
Overfull \hbox (8.73676pt too wide) in paragraph at lines 4312--4313
[]\OT1/cmr/m/n/10 Em uma ^^Sunica noite eles tin-ham de-stru-ido to-das pos-si-
```

Figura 13: Comando para execução do programa do extra 1

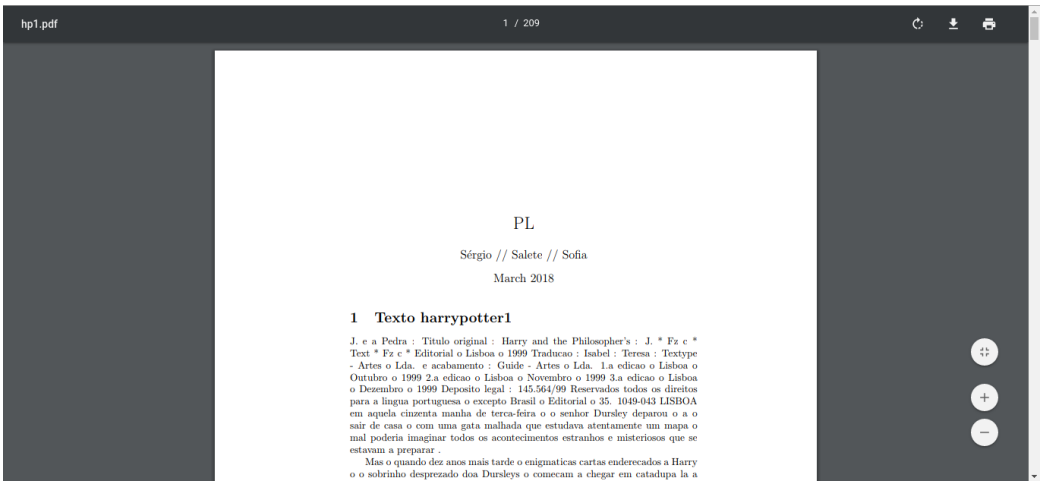

Figura 14: Respostas ao extra 1 - exemplo de ficheiro de texto em *pdf*

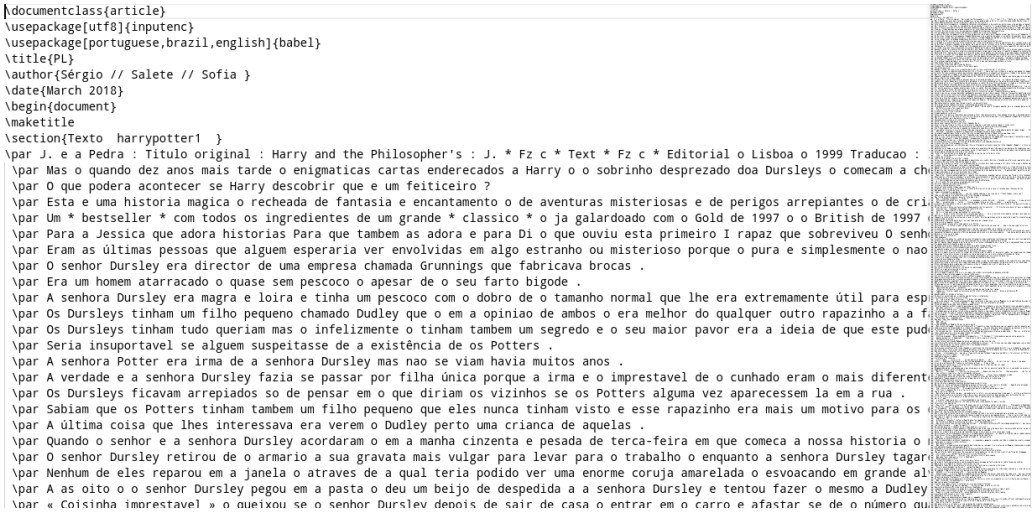

Figura 15: Respostas ao extra 1 - exemplo de ficheiro de texto

2.6 Extra 2: contar a quantidade de frases

O programa desenvolvido tem como objetivo contar o número total de frases que um documento possui.

Para tal, o programa necessita de uma variável que será incrementada sempre que for encontrado um ponto final na coluna dois, coluna que contém as palavras que juntas elaboram um texto, como se pode verificar pelo extra número um.

Desta forma, elaborou-se o seguinte programa em *awk*:

```
BEGIN {FS=" "}
$2 ~ /^\.$/ {conta++}
END {print conta}
```

Figura 16: Programa em *awk* do extra 2

Para utilizar o programa *awk* é necessário efetuar o seguinte comando:

```
sofia@sofia-X750JN ~/Desktop/PL $ gawk -f contaFrases.awk harrypotter1 harrypotter2 fl0 fl1 fl2
11812
```

Figura 17: Comando para execução do programa extra 2 e resultado correspondente

2.7 Extra 3: substituição de palavras

Este extra premite fazer a substituição da palavra *feiticeiro(s)* por *padeiro(s)*, sendo que o mesmo se aplica à versão feminina. Para além dessa palavra, substitui também *bruxa/o(s)* por *pasteleira/o(s)*.

Com esse objetivo, foi conveniente utilizar a função *gsub* que recebe duas strings e substitui todas as entradas das strings do documento que igualarem a primeira pela segunda. Deste modo, foi necessário avaliar todas as linhas do ficheiro e, sobre cada uma destas, evocar a função *gsub* duas vezes, uma para *feiticeiro* e outra para *bruxa*.

Assim sendo, o programa em *awk* foi elaborado da seguinte forma:

```
BEGIN {FS=" "; filename = ARGV[2]}
{gsub(/feiticeir/,"padeir"); gsub(/brux/,"pasteleir"); print}
END {}
```

Figura 18: Programa em *awk* do extra 3

Relativamente à execução programa *awk*, é possível após essa ação examinar o conteúdo do ficheiro gerado.

```
sofia@sofia-X750JN ~/Desktop/PL $ gawk -f padeiro.awk harrypotter1 > padeiroHP1.txt
```

Figura 19: Comando para execução do programa do extra 3

1	O	1	O
2	que	2	que
3	poderá	3	poderá
4	acontecer	4	acontecer
5	se	5	se
6	Harry_Potter	6	Harry_Potter
7	descobrir	7	descobrir
8	que	8	que
9	é	9	é
10	um	10	um
11	padeiro	11	feiticeiro
12	?	12	?

Figura 20: Respostas ao extra 3 - exemplo de ficheiro de texto gerado comparativamente ao ficheiro original

2.8 Extra 4: geração de frases

O extra quatro tem como objetivo criar uma frase a partir do dicionário desenvolvido no desafio quatro. Para este efeito, considerou-se que cada frase é composta por um determinante, um nome, um verbo e um adjetivo.

Tendo em conta a constituição do ficheiro dicionário, sabe-se que se deve procurar pela classe de cada palavra na coluna número dois e pela palavra em si na coluna três. Utilizou-se assim quatro arrays (determinante, nomes, verbos e adjetivo) e quatro inteiros que representam a posição de cada um

dos arrays e que são incrementados à medida que se adiciona informação no array correspondente. Sempre que é detetado um determinante/nome/-verbo/adjetivo, este é guardado no respetivo array. Para além disso, são gerados quatro números aleatórios que representam uma posição em cada um dos arrays. Estes números são essenciais para que a frase criada seja completamente aleatória. Deste modo, sempre que se corre o programa, a probabilidade de obter frases diferentes é bastante elevada.

Assim sendo, o programa em *awk* foi elaborado da seguinte forma:

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

$2 ~ "D" && $3 ~ /^[A-Z]/ {determinante[det++] = $3; srand(); dr = int(rand()*det)}
$2 ~ "N" {nomes[n++] = $3; srand(); nr = int(rand()*n)}
$2 ~ "V" && $3 ~ /^[a-z]/ {verbos[verb++] = $3; srand(); vr = int(rand()*verb)}
$2 ~ "AQ" && $3 ~ /^[a-z]/ {adjetivo[adj++] = $3; srand(); ar = int(rand()*adj)}

END{srand(); print determinante[dr], nomes[nr], verbos[vr], adjetivo[ar]}
```

Figura 21: Programa em *awk* do extra 4

Ao executar o programa *awk* é possível verificar qual a frase gerada por este.

```
sofia@sofia-X750JN ~/Desktop/3 ANO/PL/projeto $ gawk -f extra4.awk dicionarioHP.txt
O Percy_Weasley está azul
```

Figura 22: Comando para execução do programa do extra 4 e exemplo de um output possível do programa

2.9 Extra 5: filtragem de datas

Tendo em conta que a saga do Harry Potter representa uma história complexa que engloba vários anos, locais e personagens diferentes, o considerou-se interessante desenvolver um programa capaz de extrair todos os anos referentes à história, presentes nos ficheiros *harrypotter1* e *harrypotter2*.

Para desconsiderar os anos referentes ao lançamento dos livros ou edições, foi necessário verificar em que extrato de cada ficheiro é que começa a história em concreto. Como no primeiro ficheiro o extrato em que começa a história

é o 5 e no segundo ficheiro é o 2, foi também necessário ter em consideração qual o ficheiro que estamos a processar.

Assim, primeiro verifica-se o nome do ficheiro, utilizando-se um contador para cada ficheiro, sendo o hp1 referente ao harrypotter1 e o hp2 ao harrypotter2, e incrementa-se cada contador até ser alcançado o extrato pretendido. A partir deste momento, são verificadas todas as linhas com o intuito de encontrar os anos presentes na história. Sabendo que um ano é representado por quatro dígitos, desenvolvemos a expressão regular ∧[0-9]{4}$. Em cada linha é então verificado se a coluna dois respeita esta ER. Por último, os anos extraídos são ordenados de forma crescente e guardados em ficheiro através de um pipe.

Desta forma, elaborou-se o seguinte programa em *awk*:

```
BEGIN {FS = " "; hp1 = 0; hp2 = 0}

FILENAME == "harrypotter1" {if(hp1 < 5 && NF == 0){
                                hp1++};
                            if(hp1 == 5 && $2 ~ /^[0-9]{4}$/){
                                datas[$2]}
                            }

FILENAME == "harrypotter2" {if(hp2 < 1 && NF == 0) {
                                hp2++};
                            if(hp2 == 1 && $2 ~ /^[0-9]{4}$/) {
                                datas[$2]}
                            }

END {for(i in datas) print i  | "sort -n > datas.txt"}
```

Figura 23: Programa em *awk* do extra 5

Após a execução do programa *awk*, é possível observar os resultados gerados abrindo o ficheiro *datas.txt*.

```
sofia@sofia-X750JN ~/Desktop/PL $ gawk -f datasHP.awk harrypotter1 harrypotter2
```

Figura 24: Comando para execução do programa do extra 5

```
1289
1473
1492
1637
1709
1875
1945
```

Figura 25: Respostas ao extra 5

3 Conclusão

A elaboração deste primeiro trabalho permitiu-nos aprofundar os nossos conhecimentos relativos à linguagem *awk* e expressões regulares. Para além disso, como o grupo optou por implementar *html*, *latex* e certos *pipes* em alguns dos programas, foram também aprofundados conhecimentos relativos a essas linguagens/ferramentas.

Todos os requisitos propostos para este trabalho foram cumpridos, estando todos os desafios implementados e funcionais. É também importante referir que foram desenvolvidos com êxito programas *awk* adicionais. Esses programas extras elaborados pelo grupo permitiram explorar ainda mais os ficheiros disponibilizados, gerando novas informações.

Tendo em conta os resultados obtidos, consideramos que o trabalho foi realizado com sucesso e que foram aperfeiçoados os conhecimentos esperados.