

Processamento de Linguagens

Marcos Luís A70676

Nelson Parente A71625

8 de Março de 2018

Conteúdo

1	Introdução	3
2	Processador de transações da Via Verde	4
2.1	Análise do texto-fonte	4
2.2	Ações Semânticas	5
2.3	Estruturas de Dados Globais	5
2.4	Filtro de Texto - Sistema de Produção GAWK	5
2.4.1	Número de 'entradas' em cada dia do mês	5
2.4.2	Lista de locais de 'saída'	6
2.4.3	Total gasto no mês	7
2.4.4	Total gasto no mês apenas em 'parques'	8
3	Autores Musicais	10
3.1	Análise do texto-fonte	10
3.2	Ações Semânticas	10
3.3	Estruturas de Dados Globais	11
3.4	Filtro de Texto - Sistema de Produção GAWK	11
3.4.1	Lista total dos Cantores	11
3.4.2	Total canções do mesmo autor	12
3.4.3	Autores e seus títulos	13
4	Dicionauro	15
4.1	Análise do texto-fonte	15
4.2	Ações Semânticas	16
4.3	Estruturas de Dados Globais	16
4.4	Filtro de Texto - Sistema de Produção GAWK	16
4.4.1	Indice,Definição e Categoria Gramatical	16
4.4.2	Numero de entradas de cada Domínio	19
5	Conclusões	21

Capítulo 1

Introdução

Neste primeiro trabalho existem como objetivos previamente definidos o aumento da experiência em ambiente Linux, a pratica de desenvolvimento de expressões regulares e a utilização do sistema de produção para filtragem de texto GAWK. Foram disponibilizados quatro problemas em concreto dando aos alunos a opção de escolher um dos quatro para realizar, o grupo resolveu na totalidade três dos quatro exercícios , sendo estes , o Processador de transações da Via Verde , Autores Musicais e Dicionauro.

Para a resolução destes exercícios seria foi necessário conhecimentos referentes a expressões regulares , a linguagem gawk e a análise cuidada de cada um dos elementos a processar.

Capítulo 2

Processador de transações da Via Verde

Este tema consiste em analisar ficheiros sobre transações de clientes que utilizaram Via Verde, para tal foi nos fornecidos um texto fonte de nome viaverde.txt com as transações realizadas por um cliente.

2.1 Análise do texto-fonte

O texto fornecido para a realização desta tarefa tem na sua estrutura alguns padrões que após analisados conclui-se que iriam beneficiar e influenciar a estratégia utilizada. De seguida encontra-se um excerto do texto-fonte como ilustração e suporte para a estratégia adoptada.

<EXTRACTO id="011114056/08/2015	-		
<MES_EMISSAO>Ago-2015</MES_EMISSAO/>			
<CLIENTE id="514714936">			
...			Informação sobre o Cliente em questão
</CLIENTE>			
...	-		
<TRANSACCAO>			
...			Informação sobre 1 transação
</TRANSACCAO>	-		
<TRANSACCAO>			N transações
...			
</TRANSACCAO>			
...	-		
			Informações totais
</EXTRACTO>	-		

Como se pode verificar notou - se que havia um padrão sendo a "cabeça" do texto fonte utilizada para informação geral sobre o utilizador, o "corpo" do documento é digerido a todas as transações feitas pelo mesmo e o final do texto é utilizado para informações gerais. Quando necessário para cada uma das tarefas a realizar for utilizada uma das três partes acima discriminadas serão exploradas mais profundamente em secções a seguir.

2.2 Ações Semânticas

Após o reconhecimento feito previamente e depois de uma análise mais detalhada chegou-se à conclusão que cada linha do ficheiro continha apenas um campo de informação então decidiu-se alterar :

- **FS** (separador de campo) : por omissão é e conclui-se que seria mais útil defini-lo como `\n`
- **RS** (separador de registo): por sua vez por omissão é o `\n` decidiu-se altera-lo para a expressão `"<TRANSAC-CAO>"`
- **gsub** : quando for selecionado o campo que se pretende utilizou-se esta função para tratamento de linha retirando informação que não é necessária como tags, espaços e outros caracteres.

2.3 Estruturas de Dados Globais

Para a realização das tarefas foi necessário a utilização de várias estruturas de dados globais sendo estas:

- **mes** - variável introduzida no comando para saber determinada informação sobre o mês introduzido como parâmetro
- **valor** - variável inicializada a 0 a qual ao longo do processo vai-se incrementando um valor devolvendo no final o montante calculado.
- **entradas** - array utilizado para guardar informação que no final é iterada imprimindo como resultado

2.4 Filtro de Texto - Sistema de Produção GAWK

Depois da análise efetuada através da análise de padrões e estudo de tratamento da informação e o seu armazenamento procede-se então ao desenvolvimento de um filtro de texto utilizando o sistema de produção GAWK.

2.4.1 Número de 'entradas' em cada dia do mês

Pretende-se saber o número de entradas que o cliente fez num determinado mês , começa-se por analisar o padrão tirando partido da estrutura do ficheiro de texto fonte. Começa-se por reparar que o necessário para o desenvolvimento desta questão encontra-se apenas no corpo da função ou seja o primeiro registo não será necessário processar , de seguida cada transação tem associado a ela um campo denominado "DATA ENTRADA", sendo este o campo 2.

Sempre que aparece o campo com esta tag , trata-se da linha com o subg retirando as tags obtendo apenas a data. Após o tratamento é feito um split guardando numa variável temporário com nome data os vários campos da data.

- `data[1]` - dia
- `data[2]` - mês
- `data[3]` - ano

Comparando o valor passado como argumento com o valor que se encontra em `data[2]` se for do mesmo mês insere uma entrada no array `entradas` na posição correspondente ao dia , caso exista incrementa uma unidade. Por final imprime-se a lista dos respetivos dias em que o cliente fez transações e o numero delas. De seguida encontra-se o código, seguido da linha de comando para o mês 8 e o respetivo output.

```
comando: gawk -f entradas_mes.gawk -v mes=8 viaverde.xml (2.1)
```

```

BEGIN {
    FS="\n"
    RS "<TRANSACCAO>"
    mes
}

NR >= 2 {
    gsub("<DATA_ENTRADA>", "")
    gsub("</DATA_ENTRADA>", "")
    split($2, data, "-");
    if(mes==data[2]) entradas[data[1]]++ ;
}

END {
    print "mes " mes ;
    for(i in entradas) {print "dia " i " -> " entradas[i]} ;
}

```

Output:

```

mes 8
dia 17 -> 4
dia 18 -> 2
dia 10 -> 7
dia 11 -> 2
dia 21 -> 4
dia 13 -> 5
dia 06 -> 4

```

2.4.2 Lista de locais de 'saída'

Mais uma vez a questão é incidente na parte das transações neste caso no campo sobre o local de saída nesta questão adaptou-se novamente a mesma estratégia explicada na secção 2.4.1 mas neste caso para o campo 7 do registo pois é este em que se encontra o local de saída.

comando: `gawk -f locais_saida.gawk viaverde.xml` (2.2)

```

BEGIN {
    FS="\n" ;
    RS "<TRANSACCAO>" ;
}

NR >= 2 {
    gsub("<SAIDA>", "")
    gsub("</SAIDA>", "")
    locais[$7];
}

END {

```

```
for(i in locais) { if(i!=null) {print " Saida -> " i} }
}
```

Output:

```
Saida -> Ermesinde PV
Saida -> Aeroporto
Saida -> PQ A Sa Carn.I
Saida -> Neiva N-S
Saida -> Neiva S-N
Saida -> Maia II
Saida -> Angeiras N-S
Saida -> Valongo
Saida -> Braga Sul
Saida -> Custoias
Saida -> PQ Av. Central
Saida -> Povia S-N
Saida -> Ponte Pedra
Saida -> EN 205 PV
Saida -> Freixieiro
Saida -> Maia PV
Saida -> Lipor
Saida -> EN107
Saida -> Ferreiros
```

2.4.3 Total gasto no mês

Nesta questão será utilizada a variável referida na secção 2.3 com o nome "valor", todas as transações serão varridas comparando novamente o mês inserido com o mês na data da transação da mesma forma que foi executado no 2.4.1 caso seja igual incrementa-se o valor respetivo que já inclui iva subtraindo o campo 9 correspondente ao desconto caso haja.

comando: `gawk -f total_mes.gawk -v mes=8 viaverde.xml` (2.3)

```
BEGIN {
    FS="\n"
    RS "<TRANSACCAO>"
    mes
    valor=0 ;
}

NR >= 2 {
    gsub("<DATA_SAIDA>", "")
    gsub("</DATA_SAIDA>", "")
    split($5,r,"-");

    gsub("<IMPORTANCIA>", "")
    gsub("</IMPORTANCIA>", "")
```



```

        gsub(",",".")

        gsub("<VALOR_DESCONTO>","")
        gsub("</VALOR_DESCONTO>","")

        #print r[0] " " r[1] " " r[2] " valor " $8
        if(mes==r[2]) valor+=$8-$9 ;

    }

END    {
        print "mes " mes;
        print valor;
    }

```

Output:

```

mes 8
57.1

```

2.4.4 Total gasto no mês apenas em 'parques'

Nesta questão será utilizada a variável referida na seção 2.3 com o nome "valor", todas as transações serão varridas comparando novamente o mês inserido com o mês na data da transação da mesma forma que foi executado no 2.4.1 caso seja igual e também no campo numero 12 esteja a descrição "Parques de estacionamento", incrementa-se o valor respectivo que já inclui iva subtraindo o campo 9 correspondente ao desconto caso haja.

comando: `gawk -f mes_parques.gawk -v mes=8 viaverde.xml` (2.4)

```

    BEGIN {
        FS="\n"
        RS ="<TRANSACCAO>"
        mes
        valor=0 ;
    }

    NR >= 2 {
        gsub("<DATA_ENTRADA>","")
        gsub("</DATA_ENTRADA>","")
        split($2,r,"-");

        gsub("<TIPO>","")
        gsub("</TIPO>","")

        gsub("<IMPORTANCIA>","")
        gsub("</IMPORTANCIA>","")
        gsub(",",".");
    }

```

```

gsub("<TAXA_IVA>", "")
gsub("</TAXA_IVA>", "")

# for(i in r ) {print " -> " r[i]}
if(mes==r[2] && match($12 , "Parques de estacionamento")) valor+=$8-$9;
}

END {
  print "Valor gasto no mes " mes " em Parques";
  print valor;
}

```

Output:

```

Valor gasto no mes 8 em Parques
3.75

```

Capítulo 3

Autores Musicais

3.1 Análise do texto-fonte

Analisando os alguns dos vários ficheiros dados para a resolução deste exercício concluímos que a informação necessária esta concentrada no primeiro registo de cada ficheiro. Tendo isto apenas é dada atenção a este mesmo de modo a elaborar os pontos pedidos no enunciado.

Em seguida é apresentado o exemplo de um ficheiro que ajuda a representar a forma como foi tratado cada um dos ficheiros de extensão ".lyr".

```
title: Canta, canta amigo canta      - |
from: jj                             | N Diferentes Campos com informação essencial
author: António Macedo                _|

Canta canta amigo canta              - |
vem cantar a nossa canção            | Campo da Letra da Musica
tu sozinho não és nada                |
juntos temos o mundo na mão          _|

.
.
.
```

Sendo que para este exercício são executados todos os ficheiros de extensão

3.2 Acções Semânticas

- **FS** : Analisando o texto-fonte facilmente concluímos que os diferentes campos de informação estão separados por "\n", daí a necessidade de alterar o FS defeito.
- **RS** : Da mesma maneira que foi necessário redefinir o separador de campo , também o de registo teve esta necessidade sendo redefinido por "\n\n" dado ser o separador de cada registo de informação da letra da musica.
- **gsub** : a função gsub é utilizada de modo a limpar o texto de modo a ficar apenas a informação necessária.
- **split** : da mesma forma que o gsub o split é utilizado para tratamento de texto, sendo mais usado na separação de informação contida na mesma linha.
- **match** : esta função foi utilizada para dentro de cada registo percorrer os diversos campos à procura do necessário.

3.3 Estruturas de Dados Globais

Para a realização das tarefas foi necessário a utilização de várias estruturas de dados globais sendo estas:

- **singer** - array utilizado para guardar a a lista dos nomes dos cantores , no final é imprimida esta mesma e o seu respectivo tamanho.
- **authors** - array cujo o nome do author é o índice que armazena os respectivos títulos das suas músicas.
- **author** - variável global da alínea b , nome do autor é passado como parâmetro de modo a poder ser calculada a lista de todas as suas músicas.

3.4 Filtro de Texto - Sistema de Produção GAWK

Tal como no exercício anterior for feita uma análise cuidada ao código fonte de modo a determinar qual a melhor forma de tratar o texto de modo a desenvolver um filtro de texto através do GAWK.

3.4.1 Lista total dos Cantores

Pretende-se nesta primeira alínea calcular o total de cantores e a lista com seus nomes. Tendo isto , iremos precisar de um array para guardar os nomes dos cantores e o seu total irá ser dado pelo tamanho deste mesmo array.

Iremos começar então por percorrer todos os ficheiros mas só iremos olhar para os registos que contenham informação sobre a música , temos então a restrição `/^title:/` , ou seja , o GAWK apenas irá olhar para todos os registos que comecem pela expressão dada. Para cada registo é também verificada a extensão do ficheiro de modo a apenas tratar ficheiros de extensão `".lyr"`.

Após isso irá ser feito o matching de cada um dos campos do registo, através da função `match` e da expressão `"singer:"`, de modo a saber qual o cantor da música , quando o cantor é encontrado o campo é tratado de modo a eliminar espaços e texto redundante e finalmente inserido no array `singers`.

Quando todos os ficheiros forem processados , no campo `END` é percorrido o array `singers` imprimindo cada um dos cantores e no final o seu tamanho que representa o total de cantores.

comando: `gawk -f total_cantores_lista.gawk *` (3.1)

Código GAWK :

```
BEGIN{
FS="\n"
RS= "\n\n"
}

/^title:/{
    if(match(FILENAME,".lyr")==0){nextfile;}

    for(i=1;i<=NF;i++){
        if(match($i,"singer:")!=0) {
            gsub("singer:[^a-zA-Z]?", "");
            split($i,r,";." )
            for(j in r) gsub(/^ /,"",r[j])
            for(j in r) gsub(/ $/,"",r[j])
            for(j in r) singer[r[j]]
        }
    }
}

END {
```

```
for(i in singer) { if(i!=null) {print "Cantor -> " i} }
print "Total Cantores : " length(singer) ;
}
```

Output :

```
Cantor -> Quim Barreiros
Cantor -> Enapá2000
Cantor -> Alma Lusa
Cantor -> Madredeus
Cantor -> João Gilberto
Cantor -> Jorge Palma
Cantor -> José Barata Moura
Cantor -> João Villaret
Cantor -> Sitiados
Cantor -> Vicente da Câmara
      .
      .
      .
Cantor -> Fernando Tordo
Cantor -> Quim Tonho(?)
Cantor -> António Variações
Cantor -> António Calvário
Total Cantores : 130
```

3.4.2 Total canções do mesmo autor

Para a realização desta alínea foram utilizadas várias das estratégias anteriormente referidas , como por exemplo apenas olhar para registos começados pela string "title:", estas estratégias foram mantidas na realização desta segunda alínea e também na terceira.

Dado um autor , passado como parâmetro , é calculada a lista com todas as suas musicas e o seu número. Para cada registo é feito o match com a string "author:", que identifica o autor da musica , se o match for positivo o campo que contém o nome do autor é limpo e verifica-mos se é o mesmo autor passado como parâmetro , se o match for positivo o titulo da música é passado para o array titles.

Após a execução de todos os ficheiros no campo END o array que contém os títulos recolhidos é impresso apresentando no final o número total de musicas desse autor.

comando: `gawk -f total_cancoes_mesmo_autor.gawk -v author="Nome Autor" *` (3.2)

Código GAWK :

```
BEGIN{
  FS="\n"
  RS= "\n\n"
  author
}

/^title:/{
  if(match(FILENAME,".lyr")==0){nextfile;}
  for(i=1;i<=NF;i++){
    if(match($i,"author:")!=0) {
      gsub("author: ","");
      if(match($i,author)!=0){
        titles[$1]
      }
    }
  }
}
```

```

        }
    }
}

END {
    print "Autor " author
    for(i in titles) { if(i!=null) {print i} }
    print "Total Musicas : " length(titles) ;
}

```

Output :

```

Author Pedro Abrunhosa
title: Será
title: * Viagens
title: Se eu fosse um dia o teu olhar
title: Socorro
title: Tudo o que eu te dou
Total Musicas : 5

```

3.4.3 Autores e seus títulos

A ultima alínea deste exercício tem como objetivo para cada autor apresentar todas as suas músicas. As estratégias usadas nas alíneas anteriores foram mantidas.

Para cada registo o seu autor , se existir é adicionado como índice ao array titles e o título da música concatenado com os já presentes no array.

Finalmente no campo END são impressos no ecrã todos os elementos recolhidos separados pelo autor.

comando: `gawk -f cancoes_autor.gawk *` (3.3)

Código GAWK :

```

BEGIN{
    FS="\n"
    RS= "\n\n"
}

/^title:/{
    if(match(FILENAME,".lyr")==0) {nextfile;}

    for(i=1;i<=NF;i++){
        if(match($i,"title:")!=0) {
            gsub("title:", "");
            title = $i
        }
        if(match($i,"author:")!=0) {
            gsub("author:[ ]?", "");
            authors[$i] = title ",\n" authors[$i] ;
        }
    }
}

END{
    for(i in authors) { if(i!=null)

```

```

    {
    print "-----\n"
    print "Autor : " i "\n"
    print substr(authors[i] , 0 , length(authors[i])-2) ".\n"
    }
}

```

Output :

```

    .
    .
    .
-----

    Autor : Fernando Tordo

    Sou de outras coisas,
    Só o passado é que não,
    O bem aparecido,
    Fevereiro,
    Adeus tristeza.

-----

    Autor : Augusto Hilário

    Fado Hilário.

-----

    .
    .
    .

```

Capítulo 4

Dicionauro

Este tema consiste em ficheiros do tipo .txt que contém diferentes temas, termos associados a estes mesmo e as suas definições.

4.1 Análise do texto-fonte

Neste problema existem varios ficheiros de extensão .txt com entradas em português de termos. Cada termo tem como primeira linha a sigla PT e o seu nome, seguido de um número nao constante de linhas que contém desde de traduções do termo para outros dialectos, como tambem fins da sua utilização, definição, dominio ao qual corresponde entre outros.

```
PT fruta                -|
-catgra nf              | Nome, categoria gramatical,...
-exuso a banana é uma fruta. -|
EN fruit                |
AUDEN n                 |
FR fruit                |
AUDFR n                 | traduções
ES fruta                |
AUDES n                 |
DE frucht               |
AUDDE n                 -|
CN ??                   |
NU 10015                 | Dominio
DOM alimentação         -|
BT                       |
PART semente            |
PART casca              | Partes
PART polpa              |
POF árvore              |
UP fazer salada, sumo, doce, batido, compota, licor, bolo, gelado e tarte -|
Def algo como a banana, a maçã ou o morango, que cresce a partir de uma árvore ou planta | Def
GI 2                    |
IM 3                    |
ID manuel                -|
```


4.2 Ações Semânticas

Após o reconhecimento feito previamente e depois de uma análise mais detalhada chegou-se à conclusão que cada linha do ficheiro continha apenas um campo de informação então decidiu-se alterar :

- **FS** (separador de campo) : por omissão é espaço e conclui-se que seria mais útil defini-lo como `\n`.
- **RS** (separador de registo): por sua vez por omissão é o `\n` decidiu-se altera-lo para a expressão `".?[^AUD]?PT"`.
- **gsub** : quando for selecionado o campo que se pretende utilizou-se esta função para tratamento de linha retirando informação que não é necessária como tags, espaços e outros caracteres.

4.3 Estruturas de Dados Globais

Para a realização das tarefas foi necessário a utilização de várias estruturas de dados globais sendo estas:

- **num** - array utilizado para guardar o numero de vezes que cada dominio é referenciado na segunda alinea do problema.
- **variáveis html** - definiram-se variáveis para o tratamento da apresentação em html, tais como : enc, dominio, entrada, def, catgra, e end.

4.4 Filtro de Texto - Sistema de Produção GAWK

Tal como no exercício anterior for feita uma análise cuidada ao código fonte de modo a determinar qual a melhor forma de tratar o texto de modo a desenvolver um filtro de texto através do GAWK.

Dado o objectivo consistir na geração de vários ficheiros html foram definidas várias variáveis globais com segmentos HTML facilitaram bastante a elaboração destes ficheiros.

A primeira etapa deste exercício for criar o index HTML , para cada ficheiro de extensão ".txt" é criada uma entrada no índice , cada uma destas entradas terá a sua página HTML correspondente com os termos do seu ficheiro e , caso existam , a respectiva definição e categoria gramatical.

A primeira iteração do programa cria o ficheiro "index.html" que contém a ligação aos ficheiros HTML correspondentes a cada um dos ficheiros de texto. Seguidamente para cada um dos ficheiros ".txt" é construído o seu ficheiro HTML com a informação presente no ficheiro.

4.4.1 Índice, Definição e Categoria Gramatical

comando: `gawk -f 2html.gawk *` (4.1)

Código GAWK:

```
BEGIN{
FS="\n"
RS=".[^AUD]PT"
enc = "<html> <head> <meta charset='UTF-8' /> </head> <body>"
dominio = "<h1> Dominio : %s \n </h1>"
entrada = "<li> <b>Termo</b> [%d] : %s \n </li>"
def = "<b>Definição</b> : %s \n " ;
catgra = "<b>Categoria Gramatical </b> : %s \n<br> "
end = "</body> </html>" > "index.html"
print enc > "index.html"

fmt = "<li> <a href = '%s'> %s </a></li>\n"
tmp = ""
```

```
}
```

```
{  
if(match(FILENAME, ".txt")==0) {nextfile;}  
if(NF>3){  
a=FILENAME ;  
split(a,r, ".txt")  
if(!(r[1] in file_name)) {  
file_name[r[1]] ;  
a = r[1] ".html"  
printf( fmt , a , r[1] ) > "index.html"  
}  
}  
}
```

```
{  
if(match(FILENAME, ".txt")==0) {nextfile;}  
if(NF>3){  
a=FILENAME ;  
split(a,r, ".txt")  
  
if(tmp!=a){  
tmp=a ;  
j=1;  
print enc > r[1] ".html"  
printf(entrada , j , $1) > r[1] ".html"
```

```
for(i=1;i<=NF;i++) {  
if( (match($i, "-catgra")!=0) && (length($i) > 8) ) {  
split($i , spl , "-catgra")  
printf(catgra , spl[2]) > r[1] ".html"  
}  
}
```

```
for(i=1;i<=NF;i++) {  
if( (match($i, "Def")!=0) && (length($i) > 6) ) {  
split($i , spl , "Def")  
printf(def , spl[2]) > r[1] ".html"  
}  
}
```

```
}
```

```
else {  
j++;  
printf(entrada , j , $1) > r[1] ".html"
```

```
for(i=1;i<=NF;i++) {  
if( (match($i, "-catgra")!=0) && (length($i) > 15) ) {
```

```

split($i , spl , "-catgra")
printf(catgra , spl[2]) > r[1] ".html"
}
}

for(i=1;i<=NF;i++) {
  if( (match($i,"Def")!=0) && (length($i) > 6) ) {
    split($i , spl , "Def")
    printf(def , spl[2]) > r[1] ".html"
  }
}

}
}
}
END{
print end > "index.html" ;
for(i in file_name) print end > i ".html"
}

```

Output:

- [418-Topo-Corpo_Humano-Mao](#)
- [Botânica_planta](#)
- [Desporto_individual](#)
- [Tempo_estacao](#)
- [_cor](#)
- [atividade](#)
- [atividade_desporto](#)
- [alimento_fruta](#)
- [anatomia_corpohumano](#)
- [anatomia_corpohumano_noelia](#)
- [animal_doméstico_noelia](#)
- [botânica_árvore](#)
- [casa-Casa_de_Banho](#)
- [desporto_aquatico](#)
- [desporto_arte_marcial_e_luta](#)
- [desporto_atletismo](#)
- [desporto_colectivo](#)
- [desporto_de_Inverno](#)
- [desporto_equitacao_e_corrida_de_cavalos](#)
- [desporto_motorizado](#)
- [desporto_náutico](#)
- [desporto_radical](#)
- [desporto_termos_gerais](#)
- [jjBotanica_planta](#)
- [jjbotânica_árvore](#)
- [jjpropriedade_cor](#)
- [natacao_estilos](#)
- [propriedade_cor](#)
- [tempo_metereologia](#)
- [zoologia](#)

Figura 4.1: Index

- **Termo** : meteorologia
- Definição** : ciência que estuda o tempo e o clima
- **Termo** : atmosfera
- Definição** : camada de gases que não se vê que envolve a Terra e outros astros.
- **Termo** : atmosfera
- **Termo** : clima
- **Termo** : clima
- **Termo** : mudar de clima
- **Termo** : elemento
- **Termo** : temperatura
- **Termo** : estar calor
- **Termo** : estar frio
- **Termo** : estação
- Definição** : divisão do ano em quatro partes
- **Termo** : geada
- Definição** : camada fina de gelo que se forma quando o orvalho congela com o frio.
- **Termo** : granizo
- Definição** : água da chuva que congela e que cai em grãos brancos.
- **Termo** : neve
- Definição** : chuva que fica congelada quando o ar fica muito frio e que cai em flocos brancos
- **Termo** : trovão
- Definição** : barulho que os raios fazem quando, durante uma trovoada, caem na terra.
- **Termo** : vento
- **Termo** : névoa
- **Termo** : neblina
- **Termo** : nuvem
- **Termo** : bruma
- **Termo** : chuva
- **Termo** : aguaceiro
- Definição** : uma chuva que dura pouco tempo
- **Termo** : nevão
- **Termo** : tempestade
- Definição** : mau tempo, trovoada, vento forte e muita chuva

Figura 4.2: Meteorologia

4.4.2 Numero de entradas de cada Domínio

Nesta questão é nos solicitado que calculemos o numero de vezes que cada domínio aparece ou seja por outras palavras cada termo tem a si associado um domínio . Para o calculo total é preciso correr todos os ficheiros recebidos pela equipa docente e ir tratando cada campo domínio e inserir num array em que caso já contenha o domínio incrementado, incrementa uma unidade , caso seja novo incrementa o seu nome. Houve vários tratamentos necessários ao campo domínio antes de inseri-lo no array sendo eles, a palavra (Dom ou DOM) que antecedia o domínio e depois espaços que a linha tinha quer antes quer depois do domínio em si.

comando: `gawk -f dominios_entradas.gawk *` (4.2)

Código GAWK:

```
BEGIN{
FS="\n"
RS=".[^AUDIO]?PT"
}

{ if(match(FILENAME, ".txt")==0) {nextfile;}
for(i=1;i<=NF;i++){
split($i,r," ");
if(r[i] == "DOM" && r[2]!=null || r[i] == "Dom" && r[2]!=null) {
gsub(/ *$/, "", $i);
gsub("Dom ", "", $i);
gsub("DOM ", "", $i);
```

```

gsub(/^ /,"",$i);
num[$i]++;
}
}
}

END{
print "[Entradas | Dominio]"
  for(j in num) print (num[j] "      |" j )
}

```

Output:

```

[Entradas | Dominio]
55      |alimento
3       |
146     |desporto
3       |anatomia
9       |actividade
5       |vida
43      |tempo
1       |ciência
1       |anatomia
83      |propriedade
26      |Termos gerais
8       |Equitação e corrida de cavalos
22      |casa
1       |anatomia
5       |Desporto
1       |corpo humano
4       |árvore
2       |alimenta??o
15      |Animais dom?sticos
3       |Motociclismo
1       |zoologia
1       |seres vivos
4       |zoologia
1       |ferramenta
38      |animal
1       |medicina
30      |Atletismo
142     |botânica
167     |anatomia

```

Capítulo 5

Conclusões

Com este trabalho conseguiu-se ter-se uma visão mais aproximada da matéria lecionada, como objetivo principal a equipa docente solicitou a realização de um dos quatro exercícios tendo o grupo realizado três deles tendo como intuito praticar mais o planeamento e implementação de filtros de GAWK. Concluimos com sucesso quatro dos três exercícios propostos , isto permitiu ao grupo trabalhar com diferentes casos e distintas resoluções flexibilizando sempre o uso de expressões regulares necessárias para cada um dos casos.

Facilmente concluimos que o GAWK tem um poder de processamento enorme , conseguindo através de um número reduzido de linhas de código processar uma quantidade enorme de informação, sendo uma ferramenta extremamente útil e eficaz na recolha de informação através de filtros de texto.

Em termos das dificuldades sentidas na realização dos diversos problemas foi o "tratamento" de caracteres especiais tendo nós realizado na implementação os mesmo passos que foram feitos na aula.