Processamento de Linguagens (3º ano de MIEI)

Trabalho Prático nº2

Relatório de Desenvolvimento

Henrique Faria (a82200)

João Marques (a81826)

Nuno Rei (a81918)

28 de Abril de 2019

Resumo

O presente documento representa o relatório de desenvolvimento do segundo trabalho prático da unidade curricular *Processamento de Linguagens* do terceiro ano do curso *Mestrado Integrado em Engenharia Informática*. Este trabalho tem como objectivo a realização de programas que filtrem e transformem frases que correspondam a padrões definidos através de *Expressões Regulares*. Para tal vai ser utilizado o AWK, implementação do awk pela *GNU*, que permite uma relização mais eficiente e menos demorada de programas deste tipo em relação a programas feitos em, por exemplo, *C*.

Conteúdo

1 Introdução **3**

2 Análise e Especificação **4**
2.1 Descrição informal do problema 4
2.2 Especificação dos Requisitos 4
2.2.1 Dados 4
2.2.2 Pedidos 4

3 Concepção/desenho da Resolução **6**
3.1 Algoritmos 6
3.1.1 Pre-processamento 6
3.1.2 Processamento 6
3.1.3 Dot 7
3.2 Compilação e Execução 7

4 Codificação e Testes **8**
4.1 Alternativas, Decisões e Problemas de Implementação 8
4.2 Testes realizados e Resultados 8

5 Conclusão **10**

A Código do Programa **11**
A.1 preprocessamento.awk 11
A.2 processamento.awk 11
A.3 dot.awk 12

Lista de Figuras

2.1 Primeiro registo do formacao.csv 4

4.1 Execução do comando make 8
4.2 Imagem do grafo resultante do comando make 9

Capítulo 1

Introdução

O *GAWK* é a versão da *GNU* do programa *AWK* que foi escrito por *Alfred Aho, Peter J. Weinberger e Brian Kernighan* em meados de 1977 e é utilizado para gerar analisadores léxicos. Os ficheiros gawk possuem um conjunto de regras, isto é, pares de regras sobre número de campos de um registo ou um certo valor de um registo com código *C*, este número de registos é obtido através do *FS (field separator)* que divide uma linha de texto em *n+1* registos, sendo o primeiro registo a linha de texto completa. O *gawk* não necessita de ser compilado, pelo que o ficheiro *gawk* é executado diretamente com o comando *gawk* para procurar fazer correspondência com as regras definidas no ficheiro *gawk*, e caso corresponda executa o código associado.

O segundo trabalho prático desta unidade curricular consiste na utilização do *AWK* para a criação de um filtro de texto. O **enunciado a ser resolvido é o número dois**, *Processador de Processos de Formação*, no qual é primeiramente pretendida a realização de um novo ficheiro *CSV* mais limpo (sem campos vazios, sem linhas vazias extras, etc.) através de um pre-processamento. De seguida é feito um processamento para apresentar de forma legível informação referente ao Código, Título, Descrição, Diplomas e Notas. Para além destes dois processamentos realizados é realizado um terceiro para criar um ficheiro que pode ser processado com o programa *Dot* para realizar um grafo direcional.

Este documento vai relatar detalhadamente todas as decisões tomadas para a resolução do problema, desde o raciocínio até à implementação da solução, em que será apresentada a codificação e alguns testes que mostram o seu funcionamento.

O documento começa pela Análise e Especificação, no Capítulo 2, onde se faz uma análise detalhada do problema proposto de modo a compreender o que será lido pelo filtro de texto e formalizar uma ideia sobre os resultados que devem ser obtidos. De seguida no Capítulo 3, Concepção/desenho da Resolução, irão ser apresentados os algoritmos que foram pensados e que serão utilizados para a resolução. No Capítulo 4, Codificação e Testes, vão ser apresentadas todas as decisões tomadas para implementação, bem como problemas surgidos e possíveis alternativas para aquilo que foi codificado, e ainda testes realizados com os respectivos resultados de forma a mostrar o funcionamento da solução apresentada. Por fim no Capítulo 5 termina-se o relatório com uma síntese do que foi dito, conclusões tiradas, possíveis melhorias para trabalho futuro e principais dificuldades encontradas e como poderiam ser superadas.

Capítulo 2

Análise e Especificação

2.1 Descrição informal do problema

No problema descrito no enunciado **número dois**, *Processador de Processos de Formação* é pedido que sejam desenvolvidos um ou mais programas de filtragem de texto com recurso ao Sistema de Produção **AWK** de forma a obter o proposto pelo enunciado e em cada uma das alíneas.

2.2 Especificação dos Requisitos

2.2.1 Dados

Em conjunto com o enunciado do problema é fornecido um ficheiro em formato *CSV* que contém informação detalhada sobre a criação e gestão de cursos de formação profissional. O formato do ficheiro é o padrão para ficheiros deste tipo, isto é, cada campo é separado por um ponto e vírgula, porém contém uma particularidade em que os registos são separados por *oito ponto e vírgulas consecutivas*. Na Figura 2.1 podemos ver o primeiro registo do ficheiro que contém a informação sobre o que cada campo significa.

```
Estado;Código;Título;Descrição;Notas de aplicação;Exemplo NA;Notas de exclusão;Diplomas jurídico-administrativos REF;Diplomas
jurídico-administrativos complementar;Tipo de processo;Processo transversal (S/N);Dono do processo;Participante no processo;Tipo de intervenção
do participante;Código do processo relacionado;Título do processo relacionado;Tipo de relação entre processos;Dimensão qualitativa do
processo;Uniformização do processo;Prazo de conservação administrativa;Nota ao PCA;Justificação PCA;Forma de contagem do PCA normalizada;Forma
de contagem do prazo de cons. adm.;Destino final;Justificação DF;Notas;;;;;;;;
```

Figura 2.1: Primeiro registo do formacao.csv

2.2.2 Pedidos

De seguida apresentam-se os requisitos propostos no enunciado que a solução deve implementar:

a) Pre-processar o ficheiro *formacao.csv*, de modo a limpar as linhas vazias extra e linhas com os campos todos vazios. Adicionalmente quando campo **Estado** se encontrar vazio, deve ser colocado **NIL**. Este será o ficheiro que deve ser usado para realizar o resto do problema.

b) Contar o número de registos que apresentam **Código** numérico. Mostrar para estes o código da ação de formação, o título, descrição e notas;

c) Identificar os tipos diferentes e calcular o número de processos que existem por tipo;

d) Desenhar, em *DOT*, um grafo que relacione cada ação de formação (identificada pelo seu código) com todos os Diplomas jurídico-administrativos usados.

Capítulo 3

Concepção/desenho da Resolução

3.1 Algoritmos

Para a resolução do problema proposto foram criados três programas *AWK*:

1) **preprocessamento.awk**: Desenvolvido para realizar o pre-processamento do ficheiro *formacao.csv*, seguindo o proposto pela alínea **a** do enunciado;

2) **processamento.awk**: Desenvolvido para solucionar as alíneas **b** e **c**;

3) **dot.awk**: Contém as regras para a criação do ficheiro *dot* que contém o grafo pedido na alínea **d**.

3.1.1 Pre-processamento

Para a realização do pre-processamento do ficheiro *formacao.csv* foi desenvolvido o *preprocessamento.awk*. Começou-se por definir o **record separator** e o **field separator**, pois estes são a base de um bom programa em *AWK*. Para o **field separator** considerou-se que *";"* seria a melhor opção uma vez que se trata de um ficheiro *CSV*. Para o **record separator** reparou-se que apenas o padrão, *"\n"*, não seria oportuno no contexto do problema, uma vez que pelo primeiro registo do ficheiro *CSV* dado é mostrado que sete *";"* consecutivas indicam o fim de um registo. Como tal foi desenvolvida a expressão regular, *"(;{7,})\r\n"*, que indica que deve passar para outro registo quando encontrar sete ou mais ";"consecutivos, seguidos de troca de linha.

Já definida a separação dos campos e o separador de registos, começa a filtragem e transformação do ficheiro em que para todos os registos, tirando o primeiro que o foi removido pois apenas servia para mostrar o que era cada campo, se utiliza a função *gsub* e *sub* para remover as linhas extras, e substituir o *$1* por *"NIL"* quando este se encontra vazio. Para remover os registos com todos os campos vazios é utilizado o *gsub* com a expressão regular *"(;{27})"*, para trocar todas as ocorrências de vinte e sete ";"consecutivos, número de campos, pela *string* vazia.

3.1.2 Processamento

Depois do pre-procesamento realizado é criado um novo ficheiro *CSV* que será introduzido como input neste novo ficheiro *AWK*. Este programa lê o ficheiro dado como input e deixa apenas os registos referentes a Codigo, Título, Descrição, Notas de aplicação, Notas de exclusão, Diplomas juridico-administrativos, Código dos processos relacionados e Notas. Como extra este gera uma contagem do número de ocorrências de cada tipo de processo, colocando-a no fim do ficheiro gerado com o respetivo nome e nº de ocorrências.

3.1.3 Dot

Este é o último ficheiro *AWK* a ser executado, sendo que este vai gerar um ficheiro *DOT* para depois ser criado o grafo final. Para tal este ficheiro recebe como input o ficheiro *CSV* já pre-processado.

O **field separator** será mais uma vez o *";"* e como é utilizado o ficheiro pre-processado o separador de registo padrão é o considerado mais vantajoso para o desenvolvimento desta etapa. Para todos os registos é colocado num array indexado pelo código de formação, *$2*, um outro array com os vários diplomas, que são obtidos através de um *split* pelo caracter *"#"*. No fim é feito o print do conteúdo dos arrays no formato que será depois interpretado pelo **dot**.

3.2 Compilação e Execução

Para facilitar a execução do projeto foi criada uma makefile. O modo mais simples para executar o projeto é ter os ficheiros *makefile*, *preprocessamento.awk*, *processamento.awk*, *dot.awk* e *formacao.csv* na mesma diretoria e no terminal executar apenas *make*, para tal é necessário ter os programas *AWK* e *dot*. Quando executado desta forma são realizados todos os processamentos de cada programa. No caso de se querer verificar os ficheiros intermédios gerados basta executar *make preprocessamento* e temos acesso ao *CSV* limpo, com o comando *make processamento* obtemos uma outra tabela *CSV* que possui apenas as colunas referentes a código, notas , título e descrição bem como uma estatística do número de ocorrências de cada processo, outro comando possível é o *make grafo* que gera o ficheiro *grafo.dot* que depois é processado pelo programa *dot* para gerar a imagem final do grafo.

Capítulo 4

Codificação e Testes

4.1 Alternativas, Decisões e Problemas de Implementação

Neste trabalho para facilitar a resolução do problema foram implementados três ficheiros *AWK*, pois assim com divisão do trabalho os ficheiros ficam mais legivéis. Numa primeira fase procurou-se simplificar o ficheiro que nos foi dado, eliminando registos vazios. Posteriormente tratou-se de retirar toda a informação não referente a título, decrição, notas e códigos, bem como a criar uma estatística do número de processos de cada tipo lidos e por fim gerar um ficheiro *.dot* para criar uma imagem visual mais percetível das associações códigos-diplomas.

4.2 Testes realizados e Resultados

Mostra-se a seguir o teste realizado e os respectivos resultados obtidos:

No primeiro caso usamos o comando *make* para compilar e executar o programa por nós. Desta forma este mostra-nos a imagem final do grafo, como podemos ver de seguida.

```
henrique:TP2$ make
gawk -f preprocessamento.awk < formacao.csv > next.csv
gawk -f processamento.awk < next.csv > next2.txt
gawk -f dot.awk < next.csv
dot -Tjpeg grafo.dot -o grafo.jpeg
rm grafo.dot
henrique:TP2$
```

Figura 4.1: Execução do comando make

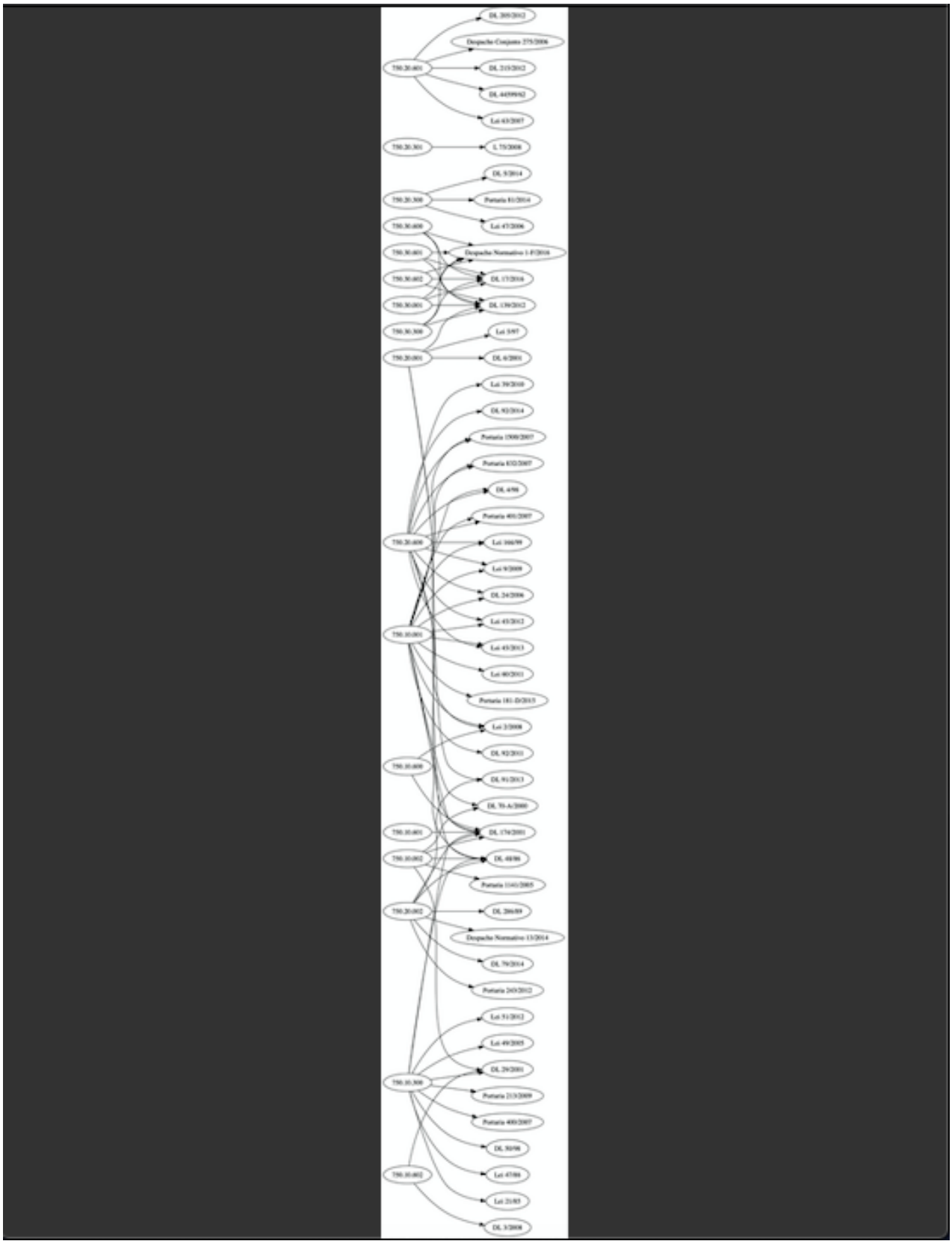

Figura 4.2: Imagem do grafo resultante do comando make

Capítulo 5

Conclusão

A avaliação feita pelo grupo quanto à solução obtida é bastante positiva, pois todos os pedidos foram concretizados.

Depois de uma breve reflexão sobre o resultado final o grupo considera que o trabalho realizado ficou dentro das espectativas, podendo proventura serem acrescentados alguns tópicos para uma maior extração de informação do ficheiro inicial. Visto que o *AWK* é uma ferramenta muito poderosa no que toca a dividir um ficheiro de texto por *RF* através de *RS* e dividir esses respetivos *RF* através de *FS* e também a verificar esse número de campos ou mesmo verificar a expressão de um deles e o que fizer *match* com essas regras criadas pelo grupo ele irá fazer alguma coisa com elas, mas caso existam informações que não sejam necessários apanhar não é preciso criar nenhuma regra específica pois este irá apenas apanhar as regras criadas.

Poderiam também ter sido criados mais ficheiros *dot* para utilização com o respetivo programa *dot* para a criação de grafos com outras informações ou mesmo gerar um grafo por cada código de formação com os respetivos diplomas.

Apêndice A

Código do Programa

A.1 preprocessamento.awk

```
BEGIN {
FS=";"
RS="(;{7,})\r\n"
}
{
gsub(/"(;{27})"/,"")
}
NR>1 {
gsub(/\n/,"")
gsub(/\r/,"")
if ($0=="")
sub($0,"")
  else
  {
  if ($1=="")
  sub($1,"NIL")
  print
  }
}
END {}
```

A.2 processamento.awk

```
BEGIN     {
        FS=";"
         RS="\n"
          }
  {
            if(NF<=7){
               b++;
               print "\n" $2 ";" $3 ";" $4 ";" $5 ";" $7 ";" $8
```

```
            a[$10]++;
          } else if(NF<=15) {
            b++;
          print "\n" $2 ";" $3 ";" $4 ";" $5 ";" $7 ";" $8 ";"  $15
          a[$10]++;
           }else if(NF <= 21)  {
            b++;
          print "\n" $2 ";" $3 ";" $4 ";" $5 ";" $7 ";" $8 ";"  $15 ";\n" $21
          a[$10]++;
           }else {
            b++;
            print "\n" $2 ";" $3 ";" $4 ";" $5 ";" $7 ";" $8 ";"  $15 ";" $21 ";\n" $27
            a[$10]++;
          }
         }
END   {
         print "\n\n Nº de ocorrencias registadas : " b;
         for(k in a){
            l = k;
            if(l == ""){
                 l="Desconhecido";
            }
            print "Processo : " l " -> " a[k]
         }
        }
```

A.3 dot.awk

```
BEGIN  {
FS=";"
print "digraph {" > "grafo.dot"
print " rankdir=LR;" > "grafo.dot"
}
NR > 0 {
if ($8!="") {
str = substr($8,2,length($8)-2)
split(str,v,"#")
for(a in v)
if(v[a]!="")
resultado[$2][v[a]]++
}
}
END  {
for(i in resultado)
for(j in resultado[i])
printf(" \"%s\" -> \"%s\";\n",i,j) > "grafo.dot"
print "}" > "grafo.dot"
```

```
}
```