Trabalho de Grupo nº 1 Sistemas Operativos

José Galinha nº 13077, Luis Adriando nº 15367

10 de Dezembro de 2020

Conteúdo

Parte I

Capítulo 1

Intrudução

O Objectivo deste trabalho é a elaboração de um conjusto de scripts utilizando as várias liguagens de script disponiveis para o sistema operativo Linux e Windows com o objetivo de desenvolver uma ferramenta de criação de dicionários para o programa Eugénio V3. O programa Eugénio V3 trata-se de uma ferramente de predição de palavras para o uso de pessoas que tenham dificuldades motoras que lhe dificultem escrever em teclado.

Parte II

Etapas de desenvolvimento

Capítulo 2

Preparação

Nesta estapa iremos usars um cript para configurar a estrutura de pastas a ser utilizada, assim como a descompreção do ficheiro do *switchboard*. Para este primeiro passo foi desenvolvido o script `main.sh`.

2.1 Ficheiro main.sh

Para facilitar uso dos vários scripts, foi criado um script `main.sh` que serve como menu para as várias opções, assim como verificação dos sistemas de pastas necessárias para o funcionamento dos scripts, criando as mesmas caso ainda não existam, assim como localiza também o ficheiro do switchboard a usar para criação dos diversos dicionários, solicitado a localixação do mesmo caso este não exista.

O script `main.sh` dispõe de três funcões principais, a *`SwitchBoard()`* que verificar a existência do ficheiro *switchboard.zip* na raiz do script, não a localizando, questiona o utilizador da localização da mesma. A função *`CheckFolderStructure()`* verifica a existência da estrutura de pastas pretendida criando a mesma caso não exista. Por fim a função *`Carregar_Script()`* é a função utilizada para chamar os vários scripts utilizados na criação dos diversos ficheiros.

O ficheiro termina com um cliclo *while* que funciona como menu do sistema com as várias opções do mesmo.

```
1  #!/usr/bin/env bash
2  # Autores: José Galinha, Luis Adriano
3  # Main script para utilização no trabalho de grupo 1 da disciplina SO
4  #
5  set -euo pipefail
6
7  # Variável que define o nome do ficheiro a localizar por defeitos
8  SWITCHBOARD_FILE="switchboard.zip"
9  # Variável que Define quais são as pastas necessárias para o funcionamento do
10 DIRS=("scripts" "corpus" "corpus_txt" "corpus_info" "words_dic" "sentences_dic")
11 FILE_LOCATED=false
12 DIR_STRUCT_OK=false
13 SCRIPT="scripts/criar_ficheiros.sh" # script usado para criar os vários ficheiro de texto
14 CORPUS_TXT='corpus_txt/switchboard.txt'
15 LIMIT=250000 # Limite de linhas na criação dos ficheiros
16 feeling="sleep 0.5"
17
18 # Função que procura o ficheiro do switchboard e se não encontrar solicita a sua
19 # localização
20 SwitchBoard(){
21     echo " Localizando o ficheiro $SWITCHBOARD_FILE"
22     $feeling
23     if [ -f $SWITCHBOARD_FILE ]; then # Verifica se o fich. se encontra na dir
24         echo -e " Ficheiro $SWITCHBOARD_FILE encontrado\n"
25         FILE_LOCATED=true
26     else
```

```
27          while :
28          do
29              echo -e " Ficheiro não encontrado\n"
30              echo "Insira o nome do ficheiro do switchboard!"
31              echo "Ficheiros .zip na directoria $PWD"
32              for file in ./*.zip # mostra todos os fich. disp. na dir.
33              do
34                  echo " -> ${file##*/}"
35              done
36              echo -e "Insira a localização do ficheiro do switchboard : "
37              read ficheiro
38              if [ -f $ficheiro  ]; then # verifica se o fich. existe
39                  clear
40                  echo -e " Ficheiro $ficheiro encontrado\n"
41                  SWITCHBOARD_FILE=$ficheiro
42                  FILE_LOCATED=true
43                  break
44              fi
45              clear # limpa a consola
46          done
47      fi
48  }
49
50  # Função para verificar se existe a estrutura de dir. a usar procura uma a uma
51  # caso não existam cria as mesmas
52  CheckFolderStructure(){
53      echo -e "\n  Incializando a verificação da estrutura de diretorias\n"
54      $feeling
55      for dir in ${DIRS[@]};
56      do
57          echo -n "  Verificando a existência da diretoria '$dir'"
58          if [ -d $dir ]; then
59              $feeling
60              echo " "
61          else
62              echo " "
63              echo -n "     diretoria inexistente criando diretoria '$dir'"
64              $feeling
65              mkdir $dir
66              [[ $? -eq 0 ]] && echo " " || echo " "
67          fi
68      done
69      echo -e "Diretorias \n"
70      DIR_STRUCT_OK=true
71  }
72
73  # Função para carregar os scripts
74  Carregar_Script() {
75      clear
76      if [ -f $1 ]; then
77          source $1 ${2:-""} ${3:-""}
78      else
79          echo -e " ficheiro '$1 não localizado\n"
80          # todo realizar o download do ficheiro quando não encontrado (git?)
81          echo -en "\npressione qualquer tecla para voltar ao menu anterior!"
82          read -n 1
83      fi
84  }
```

```
85
86  while true; do
87      clear >$(tty)
88      echo "Bem vindo ao gerador de corpus para o Eugénio V3!"
89      echo "--------------------------------------------------"
90      if ! $DIR_STRUCT_OK; then
91          CheckFolderStructure
92      else
93          echo "Diretorias "
94      fi
95      if ! $FILE_LOCATED; then
96          SwitchBoard
97      else
98          echo -e "Ficheiro do switchboard  \n"
99      fi
100     echo "Escolha a opção pretendida"
101     echo "--------------------------"
102     echo " 0 - sair"
103     echo " 1 - descompactar o switchboard"
104     echo " 2 - caracterizar o corpus utilizado (corpus_info.txt)"
105     echo " 3 - criar ficheiro de palavras (words.txt)"
106     echo " 4 - criar ficheiro de pares palavras (words_pairs.txt)"
107     echo " 5 - criar ficheiro de frases (sentences.txt)"
108     echo " 6 - criar ficheiro de pares de frases (sentences_pairs.txt)"
109     echo " "
110     read -p " -> " option
111     case $option in
112         0) break ;;
113         1) Carregar_Script "scripts/unzip_switchboard.sh" ;;
114         2) Carregar_Script "scripts/caracterizar_corpus.sh" ;;
115         3) Carregar_Script $SCRIPT "words_table.awk" "words_dic/words.txt" ;;
116         4) Carregar_Script $SCRIPT "word_pairs.awk" "words_dic/words_pairs.txt" ;;
117         5) Carregar_Script $SCRIPT "unique_phrases.awk" "sentences_dic/sentences.txt" ;;
118         6) Carregar_Script $SCRIPT "pair_phrases.awk" "sentences_dic/sentences_pairs.txt"
119     esac
120 done
121
122 echo "FIM"
```

Capítulo 3

Descompressão do corpus

Nesta espada iremos desompactar o corpus, para o efeito foi desenvolvido o script `unzip_switchboard.sh` que é chamado pelo script `main.sh` opção 1.

3.1 Ficheiro `unzip_switchboard.sh`

Neste script realizamos a extração dos ficheiros do `switchboard.zip`. O primeiro passo é usar o comando *`unzip -uqj $SWITCHBOARD_FILE "switchboard/*.txt-d corpus`* que realiza apenas a extração dos ficheiros `txt` para a pasta `corpus`, dando-nos indicação que a operação foi concluída com sucesso ou não, recebendo mensagens que indicam o caso. Seguidamente o script cria junta todos os ficheiros de texto que existem no corpus num unico ficheiro com o nome `switchboard.txt`, para hevitar erros antes de ser criado, o scirpt verifica se já existe e remove caso exista antes de criar o novo.

```
#!/usr/bin/env bash
# unzip_switchboard.sh
# Autores: José Galinha, Luis Adriano
set -euo pipefail

echo -e " Iniciando descompactação do ficheiro $SWITCHBOARD_FILE "
$feeling
# comando unzip que apenas extrai os ficheiros txt do arquivo
# https://unix.stackexchange.com/questions/59276/how-to-extract-only-a-specific-folder-from-a-zipped
unzip -uqj $SWITCHBOARD_FILE "switchboard/*.txt" -d corpus
$feeling

if [ $? -eq 0 ]; then # verifica se o comando unzip concluiu com sucesso
    echo "Ficheiros extraidos com sucesso "
    echo -n "Juntanto todos os ficheiros de texto num só  ..."
    # verificar se o fich. output já existe, se existir apaga-o
    [[ -f output ]] && rm output
    # percorre todos os ficheiros txt da pasta corpus e junta-os num fich. temp.
    for i in corpus/*.txt; do
        cat $i >> corpus/output
    done
    echo "  ficheiro criado com sucesso"
    echo -n "Movendo o ficheiro para a diretoria 'corpus_txt'"
    $feeling
    # verifica se o ficheiro já existe, se existir apaga-o
    [[ -f corpus_txt/switchboard.txt ]] && rm corpus_txt/switchboard.txt
    mv corpus/output corpus_txt/switchboard.txt
    $feeling
    # verifica se o comando mv teve sucesso, e mostra o output correspondente
    [[ $? -eq 0 ]] && echo " " || echo " "
    echo " Ficheiro criado em 'corpus_txt/switchboard.txt'"
```

```
32  else
33      echo "A extaação dos ficheiros terminou com erros "
34  fi
35
36  echo -en "\nPressione qualquer tecla para voltar ao menu anterior!"
37  read -n 1
```

Capítulo 4

Caracterização do corpus

4.1 Ficheiro `caracterizar_corpus.sh`

Neste script, queremos caracterizar o ficheiro em relação ao número de caracteres, linhas não vazias, palavras, palavras diferentes, o quociente entre o total de palavras e palavras diferentes, frases, frases diferentes e o quociente entre o total de frases e frases diferentes. Para se realizar estas diferentes tarefas foram criados diferentes scripts: um para o número de palavras, um para o número de palavras diferentes, um para as frases e um para as frases diferentes, estes scripts serão analisados mais abaixo. Estes scripts são associados a várias variáveis de modo a serem chamados quando o script principal for corrido. A primeira tarefa do script é verificar se os scripts secundários existem, se não forem o programa não corre. Após isso, o script determina o número de caracteres com o comando *number_of_chars=$(wc -m $CORPUS_TXT | $cmd '{print $1}')*, segue-se o número de frases não vazias com o comando *number_of_non_empty_lines=$(sed '/^\s*$/d' $CORPUS_TXT | wc -l)*. De seguida, o script principal corre os scripts de palavras diferentes e palavras totais e com os resultados desses scripts realiza o quociente entre eles com o comando *quociente_palavras=$(awk "BEGIN{print ($unique_words * 100) / $words}")*. Segue-se o mesmo processo com os scripts de frases diferentes e frases totais e a realização do quociente entre eles com o comando *quociente_frases=$(awk "BEGIN{print ($unique_phrases * 100_) / $phrases}")* . No final, o script cria um ficheiro output e coloca as respostas das diversas variáveis nesse ficheiro, apresentando também o resultado ao utilizador.

```
#!/usr/bin/env bash
################################################################################
#                              caracterizar_corpus.sh                          #
# Autores: José Galinha, Luis Adriano ##########################################
# Ficheiro que chama dos vários scripts para realizar a caract. do corpus #####
################################################################################

set -euo pipefail

CUW="scripts/count_unique_words.awk"
CW="scripts/count_words.awk"
CP="scripts/count_phrases.awk"
CUP="scripts/count_unique_phrases.awk"
control=true
OUTPUT_FILE="corpus_info/corpus_info.txt"


# verifica a existência do prog. nawk, em alternativa usa o awk
[[ $(command -v nawk) ]] && cmd="nawk" || cmd="awk"
if [ ! -f $CUW ]; then
    echo " O ficheiro '$CUW' não foi encontrado, o programa não pode continuar"
    control=false
fi
```

```
24
25  if [ ! -f $CW ]; then
26      echo " O ficheiro '$CW' não foi encontrado, o programa não pode continuar"
27      control=false
28  fi
29
30  if [ ! -f $CP ]; then
31      echo " O ficheiro '$CP' não foi encontrado, o programa não pode continuar"
32      control=false
33  fi
34  if [ ! -f $CUP ]; then
35      echo " O ficheiro '$CUP' não foi encontrado, o programa não pode continuar"
36      control=false
37  fi
38  if $control; then
39      echo -e "Caracterizando o corpus... \n"
40      number_of_chars=$(wc -m $CORPUS_TXT | $cmd '{print $1}')
41      number_of_non_empty_lines=$(sed '/^\s*$/d' $CORPUS_TXT | wc -l)
42      unique_words=$(cat $CORPUS_TXT | $cmd -f $CUW )
43      words=$(cat $CORPUS_TXT | $cmd -f $CW)
44      quociente_palavras=$(awk "BEGIN{print ($unique_words * 100) / $words}")
45      phrases=$(cat $CORPUS_TXT | $cmd -f $CP)
46      unique_phrases=$(cat $CORPUS_TXT | $cmd -f $CUP)
47      quociente_frases=$(awk "BEGIN{print ($unique_phrases * 100) / $phrases}")
48
49      echo "Número de caracteres: $number_of_chars" > $OUTPUT_FILE
50      echo "Número de linhas não vazias: $number_of_non_empty_lines" >> $OUTPUT_FILE
51      echo "Número de palavras: $words" >> $OUTPUT_FILE
52      echo "Numero de palavras diferentes: $unique_words" >> $OUTPUT_FILE
53      echo "Apenas $quociente_palavras% das palavras são unicas" >> $OUTPUT_FILE
54      echo "Número de frases: $phrases" >> $OUTPUT_FILE
55      echo "Número de frases únicas $unique_phrases" >> $OUTPUT_FILE
56      echo "$quociente_frases% das frases são únicas" >> $OUTPUT_FILE
57      # https://www.shell-tips.com/bash/math-arithmetic-calculation/
58      # printf %.2f%% "$((10**3 * 100 * $unique_words / $words))e-3"
59      cat $OUTPUT_FILE
60
61  fi
62
63  echo -en "\nPressione qualquer tecla para voltar ao menu anterior!"
64  read -n 1
```

Capítulo 5

Criação dos vários ficheiros

Para criação dos vários ficheiros foi desenvolvido o script `criar_ficheiros.sh` que por sua vez recebe os ficheiros de destino a gerar e o script awk responsável pelo tratamento da informação

5.1 Ficheiro `criar_ficheiros.sh`

Neste script, procuramos criar um ficheiro de palavras, contar as palavras do ficheiro e quantas vezes cada palavra ocorre. O primeiro passo é a colocar em variável o script de palavras para a criação do ficheiro de palavras e um limite. O script deve dar indicação que esta operação foi um sucesso ou não. De seguida, ocorre a criação do ficheiro de palavras com o comando `cat $CORPUS_TXT | $cmd -f $WT | sort -k 1 | $cmd -v limit=$LIMIT -f $LM > $OUTPUT_FILE`. Por fim, o ficheiro indica se a operação anterior foi completa com sucesso.

```
#!/usr/bin/env bash
# Autores: José Galinha, Luis Adriano
#
#
set -euo pipefail

WT="scripts/"$1
OUTPUT_FILE=$2
LM="scripts/limit.awk"

# verifica a existência do prog. nawk, em alternativa usa o awk
[[ $(command -v nawk) ]] && cmd="nawk" || cmd="awk"
# Verifica se o script awk existe
if [ ! -f $WT ]; then
    echo " O ficheiro '$WT' não foi encontrado, o programa não pode continuar"
else
    echo -e "Criando o ficheiro ... \n"
    # Criação do ficheiro de criação de palavras
    cat $CORPUS_TXT | $cmd -f $WT | sort -k 1 | $cmd -v limit=$LIMIT -f $LM > $OUTPUT_FILE
    # Verifica se o comando anterior foi concluido com sucesso
    [[ $? -eq 0 ]] && echo "Ficheiro '$OUTPUT_FILE' criado com sucesso " ||
            echo "Erro na criação do ficheiro "
fi

echo -en "\nPressione qualquer tecla para voltar ao menu anterior!"
read -n 1
```

5.2 Ficheiro `count_words.awk`

Neste script, procuramos o número de palavras no SwitchBoard. Para calcular isso, primeiro definimos que uma palavras como um conjunto de caracteres que não possua dois travessões seguidos com as con-

dições `( $0 ~ /[-A-Za-z]+/ ) e ( $0 !~ /\-\-/)`. Quando se deteta uma palavras ela é guardada na variável tabelaOcorrencias. No final o script indica o total de palavras encontradas.

```
#!/usr/bin/awk
################################################################################
#                                  count_words.awk                             #
# Autores: José Galinha, Luis Adriano ##########################################
# Fichiro para contar a palavras num ficheiro de texto #########################
################################################################################


BEGIN{
    RS="[ \n\t,.«»:)(;/?\"!]+";
    i=0;
}

# Para ser considerada palavra tem de ser constituida por os caracteres indicados
# && nao conter dois travessoes seguidos
( $0 ~ /[A-Za-z]+/ ) && ( $0 !~ /\-\-/) {
    if (length($0) > 1)
        i++;
}

END{
    print i
}
```

5.3 Ficheiro `count_unique_words.awk`

Neste script, procuramos o número de palavras diferentes no SwitchBoard. Para calcular isso, primeiro definimos que uma palavras como um conjunto de caracteres que não possua dois travessões seguidos com as condições `( $0 ~ /[-A-Za-z]+/ ) e ( $0 !~ /\-\-/)`. Quando se deteta uma palavras ela é guardada na variável tabelaOcorrencias e se for encontrada outra vez é incrementado um contador para indicar que tal ocorreu. No final o script indica o número de palavras únicas encontradas.

```
#!/usr/bin/awk
################################################################################
#                              count_unique_words.awk                          #
# Autores: José Galinha, Luis Adriano ##########################################
# Ficheiro para contar as palavras unicas num ficheiro de texto ################
################################################################################

BEGIN{
    RS="[ \n\t,.«»:)(;/?\"!]+";
}

# Para ser considerada palavra tem de ser constituida por os caracteres indicados
# && nao conter dois travessoes seguidos
( $0 ~ /[-A-Za-z]+/ ) && ( $0 !~ /\-\-/) {
    palavra = tolower($0)
    if (length(palavra) > 1) {
        if( palavra in tabelaOcorrencias )
            tabelaOcorrencias[ palavra ] ++;
        else
            tabelaOcorrencias[ palavra ] = 1;
    }
}

```

```
END{
    print length(tabelaOcorrencias)
}
```

5.4 Ficheiro `count_phrases.awk`

Neste script, procuramos o número de frases diferentes do SwitchBoard. O primeiro passo é criar os delimitadores de frases denominados `RS` para determinar uma frase. De seguida, eliminamos os espaços em branco para nos certificarmos que não contamos linhas vazias usando o comando `gsub(/^[[:blank:]]+/,,$0)`. Após isso, guardamos o número de frases encontradas na variável `z`. No final, o script indica o número de frases guardado em `z`.

```
#!/usr/bin/awk
################################################################################
#                                count_phrases.awk                             #
# Autores: José Galinha, Luis Adriano ##########################################
# Ficheiro para contar as frases de um ficheiro de texto #######################
################################################################################

BEGIN{
    # Delimitadores de frases
    RS="[\n\t!?,:;..{3}]+";
    z=0;
}

{
    # Vamos eliminar todos os espaços brancos no inicio das linhas com o gsub
    gsub(/^[[:blank:]]+/,"",$0)
    # Com este if estamos a garantir que não contamos linhas vazias
    if (NF > 0) {
        z++;
    }
}

END{
    print z
}
```

5.5 Ficheiro `count_unique_phrases.awk`

Neste script, procuramos o número de frases diferentes do SwitchBoard. O primeiro passo é criar os delimitadores de frases denominados `RS` para determinar uma frase. De seguida, eliminamos os espaços em branco para nos certificarmos que não contamos linhas vazias usando o comando `gsub(/^[[:blank:]]+/,,$0)`. Após este passo, contamos cada frase que encontramos e guardamos a frase numa tabela. Se uma frase for repetida, aumentamos um contador para indicar isso. No final, o script imprime a tabela de frases encontradas.

```
#!/usr/bin/awk
################################################################################
#                            count_unique_phrases.awk                          #
# Autores: José Galinha, Luis Adriano ##########################################
# Ficheiro para contar as frases unicas num ficheiro de texto ##################
################################################################################

BEGIN{
    # Delimitadores de frases
    RS="[\n\t!?,:;..{3}]+";
}
```

```
12
13  {
14      # Vamos eliminar todos os espaços brancos no inicio das linhas com o gsub
15      gsub(/^[[:blank:]]+/,"",$0)
16      #  Com este IF estamos a garantir que não contamos linhas vazias
17      if (NF > 0){
18          if ($0 in tabela)
19              tabela[$0] ++;
20          else
21              tabela[$0] = 1;
22      }
23  }
24
25  END{
26      print length(tabela)
27  }
```

5.6 Ficheiro `word_pairs.awk`

Neste script, desejamos agrupar as palavras do *SwitchBoard* aos pares. Primeiro criamos um ciclo que agrupa as palavras quando a encontra duas para seguidas numa frase. No interior desse ciclo, criamos uma condição para determinar se numa frase existem palavras, que tem de ter mais do que uma letra. Se este for o caso adicionasse um valor ao contador `tabelaPares[par]`, se não o valor desse contador é 1. No final, o script indica-nos o número de pares encontrados.

```
1   #!/usr/bin/awk
2   ##############################################################################
3   #                                  word_pairs.awk                            #
4   # Autores: José Galinha, Luis Adriano ########################################
5   # Ficheiro para contar o numero de ocorrência de pares de palavras num fich. ##
6   ##############################################################################
7
8   BEGIN{
9   }
10  # Exclui todas as palavras que estão sozinhas
11  (NF > 1){
12      # Ciclo inicia em dois pois começamos por escolher a (x-1) palavras
13      for (x = 2; x <= NF; x++) {
14          # Verifica se o $x tem mais de uma letra para ser considerado palavra
15          if (length($x) > 1) {
16              par = tolower($(x-1) " " $x)
17              if (par in tabelaPares){
18                  tabelaPares[par]++;
19              } else {
20                  tabelaPares[par] = 1;
21              }
22          } else {
23              # Aumenta o valor do x para não fazer par com letras isoladas
24              x++
25          }
26      }
27  }
28  END {
29      # Imprime para o ecrâ a tabela de pares
30      for (par in tabelaPares){
31          print par, tabelaPares[par]
32      }
33  }
```

5.7 Ficheiro `pair_phrases.awk`

Neste script para criar pares de frases, primeiro indica-se quais as condições para ser considerada uma frase, que se faz com delimitadores de frases indicados em `RS ="[\n\t!?,:;..{3}]+"`. Após isso, removemos os espaços brancos com o comando `gsub(/^[[:blank:]]+/,,$0)` e depois substituímos com o caracter "|" com o comando `gsub(/[[:blank:]]+/,"|",$0)`. Depois realizamos o par quando a condição que o counter = 0 é confirmada e verificamos se o par já existe na tabela, que se não existir adiciona-se à tabela. No final, o programa imprime a tabela com os pares de frase.

```
#!/usr/bin/awk
##############################################################################
#                                  pair_phrases.awk                          #
# Autores: José Galinha, Luis Adriano ########################################
# Ficheiro para contar o numero de pares de frases num ficheiro de texto ######
##############################################################################

BEGIN{
    # Delimitadores de frases
    RS="[\n\t!?,:;..{3}]+";
    frase = ""
    counter = 0
}

{
    # Vamos eliminar todos os espaços brancos no inicio das linhas com o gsub
    gsub(/^[[:blank:]]+/,"",$0)
    if (NF > 0){
        # Comando gsub para substituir todos os espaços por '|''
        gsub(/[[:blank:]]+/,"|",$0)
        if (counter == 0){
            par = $0
            counter = 1
        } else {
            # Construção do par
            par = tolower(par " " $0)
            # Verificação se o par já existe na tabela, se não existir inicia a 1
            if (par in tabela) {
                tabela[par] ++
            } else {
                tabela[par] = 1
            }
            # Igual o par a ultima frase para a combinar com a próxima
            par = $0
        }
    }
}

END{
    # Impressão dos resultados obtidos
    for (frase in tabela)
        print frase, tabela[frase]
}
```

5.8 Ficheiro `words_table.awk`

O script `word_table.awk`, semelhante ao script `count_words.awk` com a diferença que é criada uma tabela de ocorrências para a criação do ficheiro a usar no dicionário.

```awk
#!/usr/bin/awk
################################################################################
#                                  words_table.awk                             #
# Autores: José Galinha, Luis Adriano ##########################################
# Ficheiro para contar o numero de ocorrências de uma palavra num ficheiro ####
################################################################################

BEGIN{
    RS="[ \n\t,.«»:)(;/?\"\"!]+";
}

# Para ser considerada palavra tem de ser constituida por os caracteres indicados
# && nao conter dois travessoes seguidos
( $0 ~ /[A-Za-z]+/ ) && ( $0 !~ /\-\-/) {
    palavra = tolower($0)
    if (length(palavra) > 1 ) {
        if( palavra in tabelaOcorrencias )
            tabelaOcorrencias[ palavra ] ++;
        else
            tabelaOcorrencias[ palavra ] = 1;
    }
}

END{
    for (palavra in tabelaOcorrencias)
        print palavra, tabelaOcorrencias[palavra];
}
```

5.9 Ficheiro `limit.awk`

O script `limit.awk` é um script auxiliar criado para limitar o número de linhas a exportar na criação dos ficheiros de dicionários, sendo-lhe passada a variável `limit` definida no script `main.sh`

```awk
#!/usr/bin/awk
################################################################################
#                                  pair_phrases.awk                            #
# Autores: José Galinha, Luis Adriano ##########################################
# Script para limitar o numero de linhas dos ficheiros txt                     #
################################################################################

BEGIN{
    i = 0
}

(i < limit) {
    print $0
    i++
}

END{
}
```

5.10 Ficheiro `windows_install.bat`

O script `windows_install.bat` foi desenvolvido para quando acompanhado pelos ficheiros criados com o script `main.sh` instalar os dicionários no sistema operativo *Windows*. O memso deteta a pasta de instalação do `Eugénio` e copia os ficheiros fazendo a conversão para os nomes e tipos de ficheiro usados pelo `Eugénio`

```
::================================================================
:: Script de instalação dos ficheiros do Eugénio V3 no windows
::
::================================================================
@ECHO OFF
:: Windows version check
IF NOT "%OS%"=="Windows_NT" GOTO NotWindows
:: ### START UAC SCRIPT ###
:: https://stackoverflow.com/questions/14639743/batch-script-to-run-as-administrator
if "%2"=="firstrun" exit
cmd /c "%0" null firstrun

if "%1"=="skipuac" goto skipuacstart

:checkPrivileges
NET FILE 1>NUL 2>NUL
if '%errorlevel%' == '0' ( goto gotPrivileges ) else ( goto getPrivileges )

:getPrivileges
if '%1'=='ELEV' (shift & goto gotPrivileges)

setlocal DisableDelayedExpansion
set "batchPath=%~0"
setlocal EnableDelayedExpansion
ECHO Set UAC = CreateObject^("Shell.Application"^) > "%temp%\OEgetPrivileges.vbs"
ECHO UAC.ShellExecute "!batchPath!", "ELEV", "", "runas", 1 >> "%temp%\OEgetPrivileges.vbs"
"%temp%\OEgetPrivileges.vbs"
exit /B

:gotPrivileges

setlocal & pushd .

cd /d %~dp0
cmd /c "%0" skipuac firstrun
cd /d %~dp0

:skipuacstart

if "%2"=="firstrun" exit

:: ### END UAC SCRIPT ###

:: ### START OF YOUR OWN BATCH SCRIPT BELOW THIS LINE ###
:: Comando para permitir os caracteres unicode
chcp 1252
SET DIR=Eugénio
SET CONTROL=false

:CheckDirectory
IF EXIST "%PROGRAMFILES%\%DIR%" (
   SET DEST=%PROGRAMFILES%\%DIR%
   SET CONTROL=true
   GOTO Copy
) ELSE (
  GOTO NotFound
)
IF EXIST "%PROGRAMFILES(x86)%\%DIR%" (
```

```
    SET DEST=%PROGRAMFILES(x86)%\%DIR%
    SET CONTROL=true
    GOTO Copy
) ELSE (
  GOTO NotFound
)

IF %CONTROL% equ false (
    GOTO NotFound
)

:NotFound
ECHO "Pasta do Eugénio não encontrada! Instale o Eugénio e volte a tentar."
GOTO Exit

:Copy
ECHO Diretoria '%DEST%' detectada iniciando copia de ficheiros
COPY /y ".\words_dic\words.txt" "%DEST%\geral.pal"
COPY /y ".\words_dic\words_pairs.txt" "%DEST%\geral.par"
COPY /y ".\sentences_dic\sentences.txt" "%DEST%\geral.frs"
COPY /y ".\sentences_dic\sentences_pairs.txt" "%DEST%\geral.paf"
ECHO " Ficheiros copiados!"
pause
GOTO Exit

:NotWindows
ECHO Este ficheiro é para ser usado num SO Windows

:Exit
exit /b 0
```

Parte III

Capítulo 6

Conclusão

Com este trabalho, treinamos a utilização de comandos de Linux na linha de comandos para a realização de scripts de modo a realizar tarefas mais complexas em relação ao sistema do Eugénio. Alem disso, treinamos também a nossa habilidade e compreensão com os diversos comandos do Linux como preparação a realizar tarefas de maior dificuldade no futuro. Podemos concluir que o objetivo da realização do trabalho, os scripts, foram concluídos com sucesso.