Processamento de Linguagens

1º TRABALHO PRÁTICO

Francisco Oliveira (a78416)
Raul Vilas Boas (a79617)
Vitor Peixoto (a79175)

Março 2018

Mestrado Integrado em Engenharia Informática

Conteúdo

Capítulo 1

Introdução

No âmbito da unidade curricular de Processamento de Linguagens, foi-nos pedido para resolver um conjunto de exercícios com a finalidade de desenvolver um filtro de texto através de expressões regulares, recorrendo ao GAWK, que analise e retire informações de ficheiros de texto.

No nosso caso, foi-nos pedido para analisar alguns ficheiros de texto, anotados no formato *Freeling*.

Inicialmente foi analisada a estrutura dos ficheiros de texto que continham a informação. De seguida, com base na informação recolhida implementamos as soluções, recorrendo a expressões regulares, que nos permitissem responder aos exercícios propostos.

Neste relatório será explicado o processo de desenvolvimento e as decisões tomadas ao longo da realização do trabalho.

Capítulo 2

Textos anotados com Freeling

Neste trabalho vamos criar programas em GAWK capazes de extrair informação de ficheiros anotados em *Freeling*. Para tal analisamos os ficheiros que nos foram dados e verificamos que as colunas presentes estavam dispostas da seguinte forma: *Num*, *Palavra*, *Lema*, *POS-tag*, *POS* (classe gramatical) e *POS-tag extenso*. Esta informação irá ser bastante útil no futuro.

2.1 Número de Extratos

Nesta questão era pedido que fosse contado o número de Extratos. É-nos dito que cada extrato está separado por um linha vazia. Sabendo isto, criamos um ficheiro GAWK que conta o número de linhas vazias e no final apresenta o seu valor, pois o numero de Extratos e linhas vazias serão iguais. Apresentamos então o ficheiro GAWK com a ER (expressão regular) que calcula o número de extratos de um ficheiro.

```
BEGIN {FS=" "}

$0 ~ /^$/ { conta++ }

END { print FILENAME " : " conta }
```

Figura 2.1: *Função em GAWK que conta o número de extratos.*

Com efeito, obteve-se os seguintes resultados para os respetivos ficheiros:

Ficheiro	**Nº Extratos**
fl1	190
fl2	102
harrypotter1	5569
harrypotter2	5432

2.2 Lista de personagens e número de ocorrências

Para esta questão foi importante notar que todos os Nomes Próprios estão marcados na coluna *POS* (coluna 5) com "NP".

Sabendo isto procuramos todas as linhas cuja coluna 5 desse *match* a "NP"e guardavamos a respetiva palavra presente na coluna *Palavra* (coluna 2) num *array* e incrementavamos o seu valor para manter um contador de repetições da mesma.

Finalizamos imprimindo a lista dos Nomes Próprios e respetivo nº de repetições (já ordenado numericamente utilizando o *sort*).

```
BEGIN {FS=" "}

$5 ~ /NP/ { names[$2]++ }

END { for(i in names) print names[i], i | "sort -n -r" }
```

Figura 2.2: *Função em GAWK que calcula a lista das personagens.*

Executando este ficheiro GAWK aplicado ao ficheiro de texto *harrypotter1.txt*, através do comando "`awk -f 2.awk harrypotter1.txt`", obtivemos uma lista com as personagens e o número de ocorrências desse nome. Apresentamos a seguir um pequeno excerto das personagens com maior número de ocorrências no documento:

1143 Harry
400 Ron
337 Hagrid
241 Hermione
150 Snape
143 Dumbledore

2.3 Lista dos verbos, substantivos, adjetivos e advérbios

Nesta alínea pretende-se listar os verbos, adjetivos, substantivos e advérbios presentes e coloca-los em ficheiros.

Primeiramente identificamos a expressão regular identificadora de cada classe, que seria "V.."para verbos, "A."para adjetivos, "N."para substantivos e "R."para advérbios. No *match* dos substantivos optamos por colocar apenas "N."porque apenas existiam Nomes Próprios (NP) e Nomes Comuns (NC) pelo que era desnecessário neste caso um *match* tão estrito. Sempre que um era encontrado a respetiva palavra na coluna 2 era colocada num *array* da respetiva classe encontrada (ex: *verbs* para verbos) e aumentado o seu contador para saber o número de repetições.

Acabamos esta questão por pegar em cada lista e colocar as palavras e o seu numero de repetições, ordenados numericamente, num ficheiro como *output*.

```
BEGIN {FS=" "}

$5 ~ /V../ { verbs[$2]++ } #verbos
$5 ~ /A./  { adjectives[$2]++ } #adjetivos
$5 ~ /N./  { substantivs[$2]++ } #substantivos   aka NP&NC
$5 ~ /R./  { adverbs[$2]++ } #adverbios

END {   for(i in verbs)
            print verbs[i], i | "sort -n -r > 3verbs.result";
        for(i in adjectives)
            print adjectives[i], i | "sort -n -r > 3adjectives.result";
        for(i in substantivs)
            print substantivs[i], i | "sort -n -r > 3substantivs.result";
        for(i in adverbs)
            print adverbs[i], i | "sort -n -r > 3adverbs.result";
}
```

Figura 2.3: *Função em GAWK que calcula a lista dos verbos, adjetivos, substantivos e advérbios.*

Executando este ficheiro GAWK aplicado ao ficheiro de texto *harrypotter1.txt*, através do comando "`awk -f 3.awk harrypotter1.txt`", obtivemos quatro ficheiros com a lista ordenada das quatro classes gramaticais pedidas:

Verbos	**Substantivos**	**Adjetivos**	**Advérbios**
47 é	21 Público	22 melhor	49 mais
18 foi	20 newsletters	10 profundo	45 não
17 são	16 trabalho	6 nacional	19 também
15 tem	14 música	5 política	14 já
14 será	13 O	5 maior	11 ainda
14 está	13 cidade	5 importante	10 muito

2.4 Dicionário implícito do Corpora

A ultima questão pedia uma lista com os *Lema*, *POS* e *Palavra* derivada associada. Para começar criamos um *array* de *array's* que teria como objetivo armazenar para cada combinação de *Lema*, *POS* e *Palavra* derivada e seu numero de repetições. Nesta fase, tomamos a opção de converter todos os *Lemas* e *Palavras* para minúsculas, usando o *tolower*, para reduzir o tamanho da lista e melhorar a leitura (ex. evitando casos de 2 palavras iguais mas uma começada com maiúscula e outra com minúscula).

Depois tínhamos de mostrar os resultados pelo que escolhemos o formato *Lema-Palavra-POS-Repetições*, pois achamos que seria o mais versátil e legível. Contudo, como visível na imagem, criamos também outras opções de apresentação, talvez mais úteis noutros casos. Para a apresentação apenas fizemos 3 ciclos *for* aninhados para percorrer a totalidade do nosso *array* de *array's* e imprimir o devido conteúdo finalizando com um *sort* para colocar o resultado ordenado alfabeticamente.

```
BEGIN { FS=" " }

NF>0 {  #results[$5][$3][$2]++;
        # tolower permite uniformizar os resultados
        results[$5][tolower($3)][tolower($2)]++;
}

END {
        for(i in results)
            for(j in results[i])
                for(k in results[i][j]){
                    # printa "Lema, Palavra, POS, Repeticoes"
                    print j, k, i, results[i][j][k] | "sort";
                    # printa "Palavra, Lema, POS, Repeticoes"
                    #print k, j, i, results[i][j][k] | "sort";
                    # printa "Repeticoes, POS, Lema, Palavra" (ordenado numericamente)
                    #print results[i][j][k], i, j, k | "sort -n -r";
                }
}
```

Figura 2.4: *Função em GAWK que determina o dicionário implícito do corpora.*

Após testarmos o ficheiro GAWK aplicado ao ficheiro de texto *harrypotter1.txt*, através do comando "`awk -f 4.awk harrypotter1.txt`", obtivemos um ficheiro com o dicionário. Apresentamos a seguir alguns excertos do mesmo:

Verbos
[??:??/??/??:00.00:am] meia-noite W 11
1000 mil Z 3
(...)
abrir abras VMS 1
abrir abriam VMI 3
(...)
voltar voltes VMS 1
(...)

Capítulo 3

Conclusão

Concluídas todas as tarefas propostas, podemos dar por finalizado este trabalho prático.

Este conjunto de exercícios permitiu consolidar a matéria lecionada nas aulas da unidade curricular, mas também obter um maior conhecimento acerca do funcionamento do GAWK, para além de permitir praticar o uso de expressões regulares para resolver problemas relativos à pesquisa de informação em ficheiros de texto.

No cômputo geral, avaliamos a nossa prestação na resolução das tarefas, como positiva.