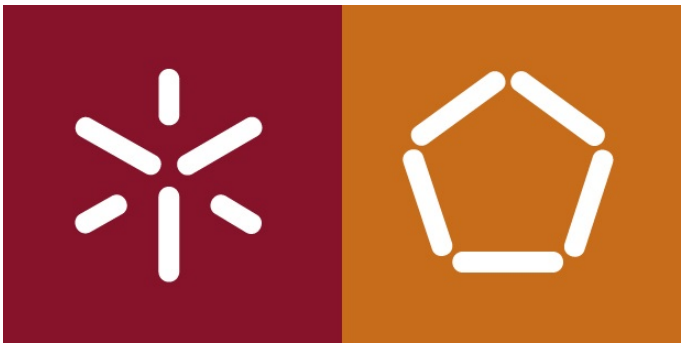

Universidade do Minho - Escola de Engenharia

Processamento de Linguagens

Processador de textos preanotados com Freeling

Autores :
Daniel Maia (A77531)

Diogo Silva(A78034)

Marco Silva(A79607)


Versão 1.0
25 de Março de 2018

Resumo

Este trabalho prático tem como principais objetivos:

- aumentar a experiência de uso do ambiente Linux e de ferramentas de apoio à programação;
- aumentar a capacidade de escrever Expressões Regulares para descrição de padrões de frases;
- desenvolver Processadores de Linguagens Regulares;
- utilizar o sistema de produção para filtragem de texto *Gawk*.

Com este intuito, serão planificados e implementados um conjunto de programas para resolver problemas envolvendo o processamento de ficheiros de texto, que requererão ao uso de cada uma destas competências.

É com esta intenção que, no final do projeto, se disponha de um conjunto de programas capazes de examinar o conteúdo de um ficheiro escrito em formato Corpora *Freeling*. A partir destes, será possível extrair o número de extratos que um dado ficheiro contém, gerar listas de nomes próprios, verbos, substantivos, advérbios e adjetivos de um ficheiro, calculando também o respetivo número de ocorrências de cada um, e determinar o dicionário implícito à sintaxe Corpora, analisando todas as palavras do texto.

Conteúdo

1 Introdução

O *AWK* é uma linguagem de programação comum em sistemas operativos *Unix*, desenhada para processamento de texto e tipicamente utilizada como uma ferramenta de extração de informação. Para tal, recorre extensivamente ao tipo de dados *string*, a *arrays* associativos e a Expressões Regulares. [1]

Um ficheiro de texto é tratado pelo *AWK* como uma sequência de registos, sendo cada linha um registo. Cada linha é subdividida num conjunto de campos (*fields*), separadas por um carater ou conjunto de carateres (por defeito, um ou mais espaços em branco), denominado *Field Separator*, ou FS. Um programa em *AWK* lê o *input* um linha de cada vez, procurando padrões definidos pelo programador e, por cada padrão que encontra, executa a ação associada.[2]

O *Gawk* (GNU *AWK*) é uma distribuição do *AWK* que permite que a extensão de funcionalidades do *AWK* através de bibliotecas partilhadas. Como tal, será esta a distribuição utilizada para este projeto, no qual se aplicará conhecimento para processar ficheiros de texto contendo *corpora* textual, especificamente, no formato *Freeling*.

Genericamente, os corpora agrupam textos aos quais adicionam informação de anotação frásica e morfossintática. O formato *Freeling* separa extratos com uma linha em branco e usa colunas separadas por espaços para a informação morfossintática de cada palavra. As colunas presentes no contexto deste projeto são: `num, palavra, lema, pos-tag, pos` (*part of speech*), `features` e `árvore`.

2 Descrição do Trabalho e Análise de Resultados

2.1 Número de extratos

Para a construção da solução que determina o número de extratos foi determinante a análise prévia dos ficheiros. Deste modo, observou-se que segundo a estruturação dos ficheiros fornecidos, cada extrato encontra-se delimitado por uma linha em branco no seu final.

```
1  A                          o                          DA0FS0  - - - - (S:0(sn:3(espec-fs:1(j-fs:1))          - - - -
2  chamada                    chamar                     VMP00SF - - - - (grup-nom-fs:3(s-a-fs:2(parti-fs:2))    - - - -
3  eleição                    eleição                    NCFS000 - - - - (grup-nom-fs:3(n-fs:3))))               - - - -
4  de                         de                         SP      - - - - (sp-de:4                                - - - -
5  Mário_Centeno              mário_centeno              NP00000 - - - - (sn:5(grup-nom-ms:5(w-ms:5))))          - - - -
6  para                       para                       SP      - - - - (grup-sp:6(prep:6)                      - - - -
7  a                          o                          DA0FS0  - - - - (sn:9(espec-fs:7(j-fs:7))               - - - -
8  chamada                    chamar                     VMP00SF - - - - (grup-nom-fs:9(s-a-fs:8(parti-fs:8))    - - - -
9  presidência                presidência                NCFS000 - - - - (grup-nom-fs:9(n-fs:9)))))              - - - -
10 de                         de                         SP      - - - - (sp-de:10                               - - - -
11 o                          o                          DA0MS0  - - - - (sn:13(espec-ms:11(j-ms:11))            - - - -
12 chamado                    chamar                     VMP00SM - - - - (grup-nom-ms:13(s-a-ms:12(parti-ms:12)) - - - -
13 Eurogrupo                  eurogrupo                  NP00000 - - - - (grup-nom-ms:13(w-ms:13)))))            - - - -
14 revela                     revelar                    VMIP3S0 - - - - (grup-verb:14(verb:14))                 - - - -
15 a                          o                          DA0FS0  - - - - (sn:16(espec-fs:15(j-fs:15))            - - - -
16 actualidade                actualidade                NCFS000 - - - - (grup-nom-fs:16(n-fs:16)))              - - - -
17 de                         de                         SP      - - - - (sp-de:17                               - - - -
18 um                         um                         DI0MS0  - - - - (sn:19(espec-ms:18(indef-ms:18))        - - - -
19 diagnóstico                diagnóstico                NCMS000 - - - - (grup-nom-ms:19(n-ms:19)                - - - -
20 feito                      fazer                      VMP00SM - - - - (s-a-ms:20(parti-ms:20)))))             - - - -
21 por                        por                        SP      - - - - (grup-sp:21(prep:21)                    - - - -
22 Boaventura_de_Sousa_Santos boaventura_de_sousa_santos NP00000 - - - - (sn:22(grup-nom-ms:22(w-ms:22))))        - - - -
23 em                         em                         SP      - - - - (grup-sp:23(prep:23)                    - - - -
24 a                          o                          DA0FS0  - - - - (sn:25(espec-fs:24(j-fs:24))            - - - -
25 década                     década                     NCFS000 - - - - (grup-nom-fs:25(n-fs:25))))             - - - -
26 de                         de                         SP      - - - - (sp-de:26                               - - - -
27 90                         90                         Z       - - - - (sn:27(numero:27)))                     - - - -
28 .                          .                          Fp      - - - - (F-term:28))                            - - - -
```

Figura 1: Extrato exemplo.

Assim, não sendo necessário efetuar qualquer ação na secção *BEGIN* do *awk* ou alterar o *field separator* uma vez que as linhas com informação não irão ser processadas neste contexto, procede-se apenas à definição de uma clausula. A condição desta será então o *NF* (*number of fields*) igual a 0, procedendo assim à contagem das linhas em branco dos ficheiros incrementando uma variável *conta*.

```
1  NF == 0 {conta++}
```

Listing 1: Condição de contagem de um extrato e respetiva ação tomada.

No final de toda a informação ser processada, na secção *END* do *awk*, procede-se à devolução do valor armazenado no contador, sendo assim apresentado ao utilizador o número de extratos presentes nos ficheiros fornecidos ao *awk*.

```
1  END {print conta}
```

Listing 2: Devolução do resultado obtido.

2.2 Lista dos personagens do Harry Potter

De forma a dar utilidade à informação contida nos ficheiros *harrypotter1* e *harrypotter2*, que se encontram preanotados com *Freeling*, tomou-se a decisão de produzir um pequeno relatório com alguma informação relacionada com as personagens presentes nos dois primeiros livros da saga, nomeadamente, **Harry Potter e a Pedra Filosofal** (HP1) e **Harry Potter e a Câmara dos Segredos** (HP2).

Deste modo, o relatório conterá um lista ordenada alfabeticamente com todas as personagens presentes nos dois livros e ainda um gráfico que compara a as 10 personagens mais frequentes no HP1 com o número de ocorrências dessas no HP2.

Assim sendo, por forma a alcançar o objetivo descrito anteriormente utilizar-se-á as ferramentas *GAWK*, *gnuplot*, *LaTeX* e alguns comandos disponíveis através do terminal linux.

2.2.1 Lista das personagens ordenada alfabeticamente

Primeiramente, é necessário extrair as personagens dos ficheiros *harrypotter1* e *harrypotter2*. Para isso, foi usada a ferramenta *GAWK*. Com base no campo *post-tag* (quarta coluna dos ficheiros - **$4**), é possível analisar o tipo das palavra associada ao registo em questão. Consequentemente, analisando as linhas com conteúdo, ou seja, com `NF > 0`, percebe-se que todos os nomes próprios começam por **NP** seguido de um conjunto de caracteres. Dado que o padrão é constante em todos os nomes próprios, utilizou-se a **expressão regular** `"^NP.*"` para fazer *match* com todos os registos e assim carregar a informação do nome de todas as personagens. Após carregar todos os nomes próprios percebeu-se que o conjunto continha muita informação incorreta. De forma a mitigar essa informação foi imposto que todos os nomes com comprimento inferior ou igual a dois (`length($2) > 2`) seriam descartados visto que não contemplam o nome de nenhuma personagem. No entanto, alguma da informação incorreta permaneceu visto ser impossível filtra-la, sendo que a única solução passaria por refazer os ficheiros iniciais *harrypotter1* e *harrypotter2*. Mais ainda, percebeu-se que muitos dos nomes próprios continham "_"em vez de um espaço. A solução passou por utilizar a função `gsub(/_/, , $2)` de forma a realizar a substituição. O nome encontra-se normalizado e pronto a ser armazenado. Para o efeito utilizou-se um **array associativo** em que o índice é a string com o nome e o valor o número de vezes que este ocorre. Além disso, é importante acrescentar que a extração das personagens só é feita a partir do registo 315 no ficheiro `harrypotter1` e do 106 no `harrypotter2`, visto que antes destes marcos as narrativas ainda não tiveram início.

```
1  personagens[$2]++
```

Listing 3: extract_pers.awk

Por fim, o array `personagens` foi percorrido de forma decrescente por valor e a informação recolhida foi escrita para ficheiro com a seguinte estrutura: FREQUÊNCIA,NOME.

```
1  PROCINFO["sorted_in"] = "@val_num_desc"
2
3  for ( i in personagens )  {
4    print personagens[i] "," i
5  }
```

Listing 4: extract_pers.awk

```
1  2571,Harry
2  1064,Ron
3  537,Hermione
4  484,Hagrid
5  280,Dumbledore
6  244,Malfoy
7  242,Snape
8  191,Lockhart
9  182,Mc Gonagall
10 177,Gryffindor
```

Listing 5: Excerto do resultado da extração das personagens.

De seguida, decidiu-se que traria significado ao relatório, que cada personagem encontrada tivesse uma pequena biografia associada. Para tal, foi usado um *dataset* [3] com a biografia de quase todas as personagens da saga *Harry Potter*. Primeiramente, o ficheiro resultante da fase de extração das personagens foi carregado para o array associativo `personagens`. Este tem como índice o nome das personagens e como valor a frequência associada.

```
1  personagens[$2] = $1
```

Listing 6: add_bio.awk

Além disso, foi carregado o *dataset* com a biografia das personagens do *Harry Potter*. O array associativo, intitulado por `dataset`, ficou com o nome da personagem como índice e o valor ficou a respetiva biografia.

```
dataset[$2] = $3
```

Listing 7: add_bio.awk

Os arrays `personagens` e `dataset` foram percorridos por índice, tomando a ordem alfabética (`PROCINFO["sorted_in"] = "@ind_str_asc"`). Cada um dos nomes encontrados são testados quanto à sua existência no dataset das biografias. Por um lado, caso exista e o `dataset` tem uma biografia correspondente, é escrito no ficheiro um registo na forma: FREQUÊNCIA,NOME,BIOGRAFIA. Caso o nome da personagem em questão faça *match* com várias personagens do array `dataset`, então é escrito o mesmo nome com várias biografias diferentes. Esta foi uma decisão de implementação tomada de forma consciente. Efetivamente, o nome das personagens extraídas dos documentos anotados com *Freeling* carece de contexto. A título exemplificativo, o nome *Weasley*, que ocorre nos dois livros, pode representar qualquer um dos membros da família *Weasley*, sendo que apenas uma abordagem semântica permitiria especificar a personagem propriamente dita. Desta forma, optou-se por registar todas as possíveis interpretações de um determinado nome, com as diferentes biografias possíveis. No entanto, caso esse não seja o efeito desejado basta trocar a ordem com que o *match* é realizado, ou seja, de `pers_name_dataset ~ pers_name` para `pers_name ~ pers_name_dataset`, no ficheiro `add_bio.awk`. Por outro lado, caso não exista uma biografia disponível para o nome próprio a ser processado então esse campo não é incluído no registo final, consequentemente apresentando-se na forma: FREQUÊNCIA,NOME.

```
58,Weasley,Father of the Weasleys and member of the Order of the Phoenix.
58,Weasley,Oldest son of Arthur and Molly. Husband of Fleur.
58,Weasley,Second son of Arthur and Molly. Works with dragons in Romania.
58,Weasley,Identical twin with George and co-owner of Weasleys' Wizard Wheezes
58,Weasley,Identical twin with Fred and co-owner of Weasleys' Wizard Wheezes
58,Weasley,Marries Harry Potter and only daughter of Molly and Arthur.
58,Weasley,Wife of Arthur and mother of the Weasleys. Kills Bellatrix.
58,Weasley,Third son of Arthur and Molly. He is a Gryffindor prefect.
58,Weasley,Harry's best friend. Marries Hermione.
```

Listing 8: Repetição da biografia para um mesmo nome

2.2.2 Gráfico comparativo do número de ocorrências das personagens

Estando a lista das personagens pronta para ser inserida no relatório torna-se necessário produzir o gráfico que compara as dez personagens com um maior número de ocorrências no HP1 com o respetivo número de ocorrências do HP2.

Para tal, são extraídas os nomes próprios dos dois livros, através do mesmo programa *GAWK* utilizado na listagem das personagens, mas desta vez o resultado é colocado em ficheiros separados. Assim sendo, é necessário converter os ficheiros resultantes, que se encontram na forma **FREQUÊNCIA,NOME**, para o formato **"NOME"FREQUÊNCIA_HP1 FREQUÊNCIA_HP2**. Este é o formato utilizado pelo *gnuplot*, ferramenta escolhida para produzir a imagem *png* com o gráfico final. Desta forma, são carregados os dez nomes mais frequentes do HP1, para um array associativo `pers_hp1`, sendo o índice o nome das personagens e o valor a frequência com que estas aparecem.

```
ARGIND == 1 && NR < 11 { pers_hp1[$2] = $1 }
```

Listing 9: convert_data.awk

Os registos do HP2 são todos carregados. A variável que tem vindo a aparecer repetidamente, de nome, `ARGIND` indica o número do argumento que está a ser processado no momento e como tal permite diferenciar o *parse* realizado consoante se trate do HP1 ou HP2.

```
ARGIND == 2 { pers_hp2[$2] = $1 }
```

Listing 10: convert_data.awk

Por forma a obter o formato desejado basta percorrer os valores do `pers_hp1` de forma descendente. Para cada instância é confirmada a existência de um mesmo nome em `pers_hp2`.

Caso a resposta seja afirmativa é impresso o registo na forma: "NOME"FREQUÊNCIA_HP1 FREQUÊNCIA_HP2. Caso contrário a componente do array `pers_hp2` torna-se nula, resultando na seguinte sintaxe: "NOME"FREQUÊNCIA_HP1 0.

```
PROCINFO["sorted_in"] = "@val_num_desc"

for (i in pers_hp1) {
  if (i in pers_hp2)
    print "\"" i "\"", pers_hp1[i], pers_hp2[i]
  else
    print "\"" i "\"", pers_hp1[i], 0
}
```

Listing 11: convert_data.awk

Por último, para produzir o gráfico basta correr o comando `gnuplot plot.dat`. Este executa os comandos presentes no ficheiro `plot.dat` que posteriormente indicará qual o ficheiro de dados que deve ser carregado para produzir o gráfico, ou seja, o `hp.dat`. Desta forma, obtém-se um gráfico de barras que compara o número de ocorrências das personagens entre os dois primeiros livros da saga *Harry Potter*.

2.2.3 Produção do ficheiro *tex* para geração do relatório final

Neste ponto tem-se um ficheiro com os nomes ordenados alfabeticamente com as respetivas frequências e a imagem em formato *png* do gráfico prontos. Neste momento, é executado o programa `export_latex.awk` que escreve num ficheiro *tex* o código necessário para incluir as duas componentes anteriormente produzidas. Além disso, associada à lista das personagens encontra-se um pequeno índice por forma a navegar de forma mais otimizada nos nomes. Este foi conseguido através do uso de dois comandos do *latex*, `hyperlink` e `hypertarget`. Efetivamente, foi criado um array associativo (`chars`) que contém como índice uma letra do abecedário e como valor a string vazia. Desta forma, sempre que surge um nome é verificado se a primeira letra deste já se encontra no array. Em caso negativo ela é acrescentada ao array e uma marca da hiperligação (*hypertarget*) é colocada na posição respetiva do ficheiro *latex*.

Finalmente, apenas resta compilar o ficheiro *tex* e gerar o pdf final. Ação conseguida através do comando: `pdflatex lista_pers_harry_potter.tex`. Como resultado é produzido o ficheiro `lista_pers_harry_potter.pdf` com todas as componentes aqui desenvolvidas integradas num único ficheiro.

Conteúdo

1 Introdução

Este documento é o resultado do processamento, através da ferramenta *GA* de textos preanotados com Freeling. Efetivamente, foram usados os dois pri ros livros da saga *Harry Potter*, nomeadamente o **Harry Potter e a P Filosofal** e o **Harry Potter e a Câmara dos Segredos**.

2 Número de ocorrências das personagens

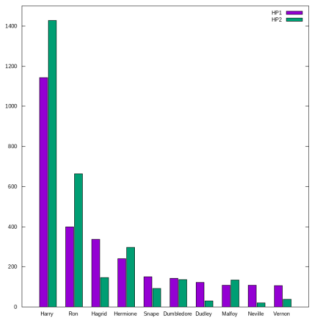

Figura 1: Comparativo do número de menções das diversas personagens e os livros "Harry Potter e a Pedra Filosofal"(HP1) e "Harry Potter e a Câ dos Segredos"(HP2).

O comparativo presente no gráfico permite concluir informações basta curiosas. Nomeadamente, que a personagem extitHarry Potter é mais me nada no segundo livro. Mais ainda no primeiro livro a personagem *Hagri* terceira mais frequente, no entanto no segundo livro esta passa a ser a *Herm* O mesmo acontece com o extitSnape e o *Dumbledore*, respetivamente.

3 Lista das personagens presentes nos livros

A lista das personagens que se encontra nesta secção é o resultado do parse dos documentos anotados com *Freeling*. No entanto, estes documentos não estão corretos na sua totalidade e como tal existe algumas palavras na lista abaixo que não são personagens, mas que foram consideradas como nome próprio. Nestes casos torna-se impossível resolver os conflitos em questão e como tal encontram-se na lista final.

3.1 Índice

3.2 Lista das personagens ordenada alfabeticamente

- **A Câmara** [1]
- **AAAAAAAAHHHH** [1]
- **AAAAAAARCH** [1]
- **AARGH** [1]
- **AL Harry** [1]
- **AS MOTOS NÃO VOAM** [1]
- **Aaargh** [1]
- **Abominável Homem das Neves** [2]
- **Achaste** [1]
- **Acho** [1]
- **Acidente** [1]
- **Adalbert Waffling** [1]
- **Adeus** [1]
- **Adeusinho** [1]
- **Adrian Pucey** [4]
- **After-Eight** [1]
- **Agarra** [1]
- **Agrippa** [3]
- **Ahem** [1]
- **Ainda** [2]
- **Ala** - Retired auror and member of the order of the Phoenix. Killed by Voldemort. [1]

Figura 2: Relatório final.

2.3 Lista dos verbos, substantivos, adjetivos e advérbios

Para a elaboração de uma lista dos verbos, substantivos, adjetivos e advérbios foi necessária mais uma vez uma análise em detalhe dos *snapshots* disponibilizados.

Deste modo, observou-se que cada uma das categorias de palavras acima descritas se encontra definida segundo a sua classe gramatical na 3ª coluna de cada um dos extratos a analisar. Por consequência, procedeu-se ao desenvolvimento de uma expressão regular para cada um dos casos acima apresentados complementando sempre com a restrição de que o *number of fields* terá de ser superior a 0, evitando assim erros no processamento da informação. Ainda dentro desta fase da análise da informação presente nos extratos, procede-se também ao tratamento da mesma, efetuando a substituição em cada um dos campos necessários dos caracteres _ por espaços, tornando assim a sua leitura mais natural.

```
1  NF > 0 && $4 ~ /^V.*/ {
2          gsub(/_/, " ", $3);
3          gsub(/_/, " ", $2);
4          verbos_lema[$3]++;
5          verbos[$3][$2]++;
```

```
    }
```

Listing 12: Preenchimento das estruturas de dados com o número de ocorrências de lemas e palavras.

Após todos os ficheiros serem processados e as frequências armazenadas nos respetivos arrays, procede-se então ao tratamento desta mesma informação para a sua apresentação.

Para uma melhor interpretação dos resultados obtidos, apresenta-se para além da listagem de todas as ocorrências de palavras e respetivo lema, um top 10 baseado no número de ocorrências. A construção deste passa por várias fases que irão ser descritas a seguir.

Primeiramente, são preenchidas estruturas auxiliares que irão albergar a informação respetiva aos tops de cada uma das categorias com as primeiras 10 ocorrências, sendo estas a informação base de comparação com os restantes resultados. Abaixo podemos ver o algoritmo utilizado para o caso das estruturas auxiliares de verbos, sendo o processo análogo para as restantes classes de palavras

```
a = 0;
      for (i in verbos_lema){
          for (j in verbos[i]){
              if (a < 10){
                  top_verbos_palavra[a] = j;
                  top_verbos_lema[a] = i;
                  top_verbos[a] = verbos[i][j]; a++;
              }
              else break;
          }
      }
```

Listing 13: Preenchimento inicial das estruturas de dados dos tops.

De seguida, as primeiras 10 ocorrências inseridas inicialmente serão ordenadas pelo número de ocorrências para que mais tarde, quando a restante estrutura de dados for percorrida, seja possível ser feita uma inserção ordenada no array top, evitando assim travessias adicionais quer das estruturas de dados extraídos, quer do próprio array de top. Na imagem apenas se encontra representado o caso de ordenação do array de verbos uma vez que para as restantes classes o processo é análogo.

```
for (d = 0; d < 10; d++){
      for (b = d; b < 10; b++){
          if (top_verbos[d] < top_verbos[b]){
              temp = top_verbos[d];
              top_verbos[d] = top_verbos[b];
              top_verbos[b] = temp;

              temp = top_verbos_lema[d];
              top_verbos_lema[d] = top_verbos_lema[b];
              top_verbos_lema[b] = temp;

              temp = top_verbos_palavra[d];
              top_verbos_palavra[d] = top_verbos_palavra[b];
              top_verbos_palavra[b] = temp;
          }
      }
  }
```

Listing 14: Ordenação inicial da estrutura de dados top.

Finalmente, são percorridas as estruturas de dados de cada uma das classes de palavras e, para cada uma das ocorrências, é verificado se essa mesma ocorrência se enquadra no cenário de top. Esta verificação consiste na tentativa da inserção da ocorrência efetuando a comparação com os dados presentes no momento no *array* top e, no caso de esta ocorrência for maior do que algum dos dados já presentes no top, os elementos com frequência menor são movidos uma posição para a direita, sendo a ultima posição do *array* descartado, e é inserida a nova ocorrência no local correto.

Apenas é possível recorrer a este método de inserção uma vez que o array inicial se encontrava já ordenado.

```
for (n in verbos_lema){
          for (o in verbos[n]){

              for(i = 0; i < 10; i++){
                  if (top_verbos[i] < verbos[n][o]){
                      #SHIFT PARA A DIREITA
                      for(j = 9; j < i; j--){
                          top_verbos[j-1] = top_verbos[j];
                          top_verbos_lema[j-1] = top_verbos_lema[j];
                          top_verbos_palavra[j-1] = top_verbos_palavra[j];
                      }
                      top_verbos[i] = verbos[n][o];
                      top_verbos_lema[i] = n;
                      top_verbos_palavra[i] = o;
                      break;
                  }
              }

          }
      }
```

Listing 15: Tentativa de inserção ordenada no *array* top de cada uma das ocorrências de palavra.

Para além dos tops descritos acima, será também apresentada a lista com todas as ocorrências, respetivo lema e frequência, ordenados alfabeticamente, para que o utilizador possa ter acesso ao relatório completo do número de ocorrências. Estas listas encontram-se também divididas segundo as classes acima referidas (verbos, adjetivos, advérbios e substantivos).

Para a apresentação desta lista completa de ocorrências, são utilizadas mais uma vez as estruturas de dados preenchidas inicialmente organizadas por lema e palavra mas, agora ordenadas alfabeticamente. Para esta ordenação, recorreu-se à função *asorti*, que ordena *arrays* com base no índice dos mesmos, como podemos ver abaixo.

Por forma a que a informação fosse apresentada da forma mais clara possível, o formato escolhido para a exportação dos dados foi o *HTML*.

Para a construção dos ficheiros finais, é necessário seguir as regras de estruturação de um ficheiro deste formato. Para isso, irão ser explicitados os passos tomados.

Ainda na secção *BEGIN*, é impressa a estruturação inicial do documento, bem como a definição do tipo de caracteres a utilizar, de modo a uniformizar o ficheiro. É definido também um titulo identificativo da classe da informação representada no ficheiro.

```
print "<html>\n<head><meta charset=\"UTF-8\"></head>\n<body>\n\n<h1 align=""\"
 center\""">Verbos</h1>" > "verbos.html"
```

Listing 16: Inicialização da estruturação dos documentos *HTML*.

Na próxima escrita para os ficheiros *HTML*, são inicializadas as tabelas de tops e respetivos títulos, como podemos ver abaixo.

```
print "<h2 align=""\"center\""">Top 10 Ocorrencias</h2>" > "verbos.html";
  print "\n<table align=""\"center\""" border='3'>\n\t\t" > "verbos.html";
  print "<tr bgcolor=""\"grey\""">\n\t\t<th>Lema</th><th>Palavra</th><th>
  Ocorrencias</th>\n\t</tr>" > "verbos.html"
```

Listing 17: Inicialização da tabela e respetivo título.

Após toda a construção dos tops descrita acima, procede-se à exportação dos mesmos para a estrutura da tabela. Neste momento podemos já fechar a tabela dos tops, uma vez que já foi exportada toda a informação necessária.

```
for (b = 0; b < 10; b++){
            #print top_verbos[b], top_verbos_lema[b], top_verbos_palavra[b];
            print "<tr>\n\t<td>" top_verbos_lema[b] "</td><td>"
  top_verbos_palavra[b] "</td><td>"top_verbos[b]"</td>\n</tr>\n" > "verbos.html";
```

```
4       }
5
6       print "\t</table>" > "verbos.html";
```

Listing 18: Exportação dos dados dos verbos para a tabela top e exportação das tags de fecho da mesma. Análogo para as outras componentes.

De seguida, procede-se à inicialização da tabela que albergará toda as palavras e informação associada adicionando um título informativo e finalmente inicializando a tabela.

```
1       print "<h2 align=""\"center\""">Todas as ocorrencias de verbos</h2>" > "verbos.
        html";
2
3       print "<table bgcolor=""\"#f2f2f2\""" align=""\"center\""" border='3'>" > "
        verbos.html"
4
5       print "\t<tr bgcolor=""\"grey\""">\n\t\t<th>Lema</th>\n\t\t<th>Palavra</th>\n\t
        \t<th>Ocorrencias</th>\n\t</tr>" > "verbos.html"
```

Listing 19: Inicialização da tabela completa de ocorrências para o caso específico dos verbos. Análogo para as outras componentes.

Como foi referido acima, neste momento procede-se à exportação de cada uma das ocorrências para a respetiva tabela com o seguinte código.

```
1       asorti(verbos_lema, verbos_lema_sort);
2       for (i in verbos_lema_sort){
3               asorti(verbos[verbos_lema_sort[i]], verbos_palavras);
4               print verbos_lema_sort[i], ":" > "verbos.txt";
5
6               for (a in verbos_palavras) conta++;
7
8               #HTML
9               print "\t<tr bgcolor=""\"#bfbfbf\""" align=""center""">\n\t\t<td rowspan
        ="conta+1">"verbos_lema_sort[i]"</td>\n\t</tr>" > "verbos.html";
10
11              for (j in verbos_palavras){
12                      print  verbos_palavras[j], " -> ", verbos[verbos_lema_sort[i]][
        verbos_palavras[j]] > "verbos.txt";
13                      #HTML
14                      print "\t<tr>\n\t\t<td>"verbos_palavras[j]"</td>\n\t\t<td>"
        verbos[verbos_lema_sort[i]][verbos_palavras[j]]"</td>\n\t</tr>" > "verbos.html"
        ;
15              }
16              conta = 0;
17      }
```

Listing 20: Exportação de cada uma das ocorrências de palavra e respetivas informações associadas para a tabela completa de palavras.

Para concluir, apenas será necessário fechar a tabela de ocorrências procedendo do seguinte modo.

```
1       print "</table>\n</body>\n</html>" > "verbos.html";
```

Listing 21: Fecho da tabela que alberga a totalidade da informação.

2.4 Dicionário implícito no CORPORA

Pretende-se extrair dos ficheiros de texto uma lista de todos os seus `lemas` e, para cada `lema`, criar uma sub-lista de `palavras` e respetivos `pos` de modo a gerar um dicionário. Como tal, observando a estrutura dos dados, observa-se que se irá guardar informação dos segundo, terceiro e quarto campos.

Neste caso o programa não necessita efetuar quaisquer ações previamente à leitura dos ficheiros, logo, não é executada nenhuma ação na secção *BEGIN*.

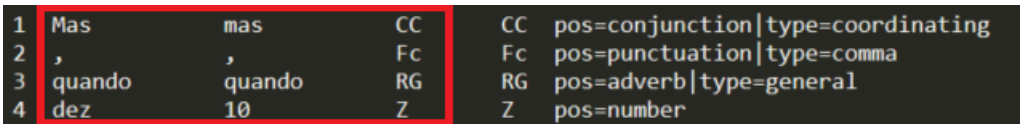

Figura 3: Os campos que serão lidos serão os em destaque.

Para que não ocorram erros de leitura de campos durante a execução do programa devido a linhas vazias, será necessário adicionar a condição de execução `NF > 0` em cada linha do ficheiro. Como se trata de um dicionário, será necessário filtrar todos os `lemas` correspondentes a pontuação (i.e. pontos, vírgulas, etc). Uma análise do formato *Freeling* mostra que todos estes têm o `pos` **F** seguido de um ou mais carateres. [4] Como tal, outra condição de processamento de informação de cada linha será `$4 !~F.+ .` Quando é encontrada uma linha que passe a condição, o seu `lema` é guardado num *array* associativo **dic_palavras**, cujos índices são os próprios `lemas`. Cada `palavra` e `pos` de um determinado `lema` será, por sua vez, guardado num *array* associativo bidimensional **dicionario**, cujos índices são o dado `lema` e a concatenação da `palavra` e `pos`, separados pelo carater *SUBSEP*, nativo ao *Gawk*.

No final da leitura dos ficheiros, procede-se à organização e armazenamento dos dados num novo ficheiro. É efetuado um *asorti* do *array* *dic_palavras* de modo a obter uma lista dos `lemas` ordenada alfabeticamente. Percorrendo o *array* resultante **palavras**, efetua-se um *asorti* a cada elemento `dicionario[palavras[i]]` para ordenar cada uma das `palavras` derivadas do respetivo `lema` por ordem alfabética, guardando-a no *array* **definicoes**. Finalmente, efetua-se um *split* de cada elemento de *definicoes* para recuperar as `palavras` e `pos`.

Tendo planificado como se iria processar os dados, ponderou-se qual seria o melhor método de armazenamento dos mesmos. Decidiu-se guardar os dados do dicionário *Corpora* em formato XML devido à sua capacidade de descrever informação com exatidão e sem ambiguidade e o facto de que o seu conteúdo é independente de formatação. Para tal, é necessário definir a estrutura do ficheiro que se pretende gerar. O ficheiro XML resultante será composto por uma *tag* **dicionario**, que contém um conjunto *tags* **termo**. Cada uma destas é composta por uma *tag* **lema** que, como o nome sugere, contém um `lema` como atributo **id**, bem como um conjunto de *tags* **palavra** que, por sua vez, guardam uma `palavra` como atributo **id** e o correspondente `pos`, na sua própria *tag*.

De modo a manter a sintaxe do XML, é necessário certificar que não são passados carateres que possam ser mal interpretadas quando lidas após a sua geração. Neste caso, trata-se apenas do carater &, que terá de ser substituído por "&". Como tal, antes da escrita de qualquer `lema` ou `palavra`, será efetuado um *gsub* para certificar que a semântica do XML permaneça intacta.

Por fim, é de notar que a primeira linha de um ficheiro XML especifica o seu *encoding*. Assim, a primeira linha escrita no ficheiro será a seguinte: $<?xml version = "1.0"?>$.

Deste modo, o ficheiro resultante toma este aspeto:

Figura 4: Exemplo do conteúdo do ficheiro resultante da execução do programa.

3 Conclusões e Sugestões

Este projeto contempla inúmeras funcionalidades que o *GAWK* fornece. Com ele foi possível processar registos de ficheiros, produzir ficheiros *tex*, converter o formato dos dados, imprimir *html* ou mesmo *XML*.

Desta forma, com a ajuda da ferramenta mencionada anteriormente conseguiu-se produzir um relatório que analisa com algum pormenor as personagens dos dois primeiros livro da saga *Harry Potter*. Mais ainda, foi possível extrair todos os verbos, adjetivos, substantivos e advérbios dos cinco ficheiros disponíveis e produzir um ficheiro *html* de melhor compreensão e fácil análise. Além disso, foram aglomerados os lemas, palavras e o pos de cada palavra num ficheiro *XML*, dicionário pronto a usar para trabalhos futuros.

No entanto, a informação presente nos documentos disponibilizados é bastante abrangente e diversa. Como tal, as funcionalidades aqui apresentadas podem ser melhoradas, ou novas podem ser acrescentadas, em iterações futuras do projeto. Desta forma, possíveis melhoramentos passariam por filtrar possíveis erros que existam no ficheiros base, para que esses não sejam carregados para o programas *AWK*. Além disso, o XML produzido pode ser integrada numa componente gráfica, de forma a providenciar ao utilizador uma melhor interface na leitura do dicionário.

Neste projeto fica evidente o processo no qual, tendo ficheiros anotados com *Freeling*, é possível extrair informação destes através de *GAWK* e por fim converte-la para praticamente qualquer formato existente, nomeadamente, *html*, *LaTeX* e *XML*.

Referências

[1] Applying Minilanguages; Case Study: awk http://www.faqs.org/docs/artu/ch08s02.html#awk

[2] The A-Z of Programming Languages: AWK https://www.computerworld.com.au/article/216844/

[3] Potter Network: characters dataset, https://github.com/efekarakus/potter-network/blob/master/data/characters.csv

[4] FreeLing 4.0 User Manual https://www.gitbook.com/book/talp-upc/freeling-4-0-user-manual/details