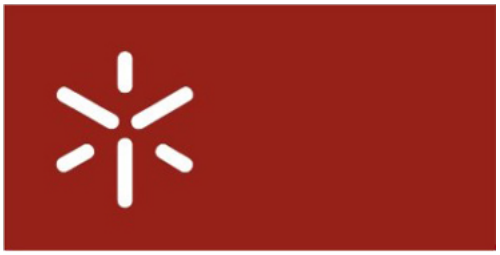

Processamento de Linguagens

Engenharia Informática

Universidade do Minho

Trabalho Prático No 1 (GAWK)

Sara Pereira
A73700

Pedro Almeida
A74310

Frederico Pinto
A73639

24 de Março de 2018

Conteúdo

1 Contextualização

No âmbito da unidade curricular de Processamento de Linguagens, foi-nos apresentado um enunciado que contém cinco problemas diferentes para resolver, sendo um deles de caráter obrigatório. Assim, depois de aplicada a fórmula dada $((73639\%5) + 1 = 5)$, verificamos que o problema a resolver pelo nosso grupo seria o problema cinco, um *Processador de Thesaurus*. Um thesaurus é um livro que relaciona palavras e conceitos, podendo ser considerado um dicionário ou uma enciclopédia. Além do exercício obrigatório, decidimos complementar o projeto com a execução do problema dois e apresentando todos os resultados do problema cinco em *HTML*.

2 Processador de Thesaurus 2

2.1 Enunciado 2.5

Em **natura.di.uminho.pt/ jj/pl-18/TP1/THE/** encontramos ficheiros .mdic que descrevem numa sintaxe simples as entradas (triplos termo1, rel, termo2) de um Thesaurus que se pretende criar automaticamente. Assim, pretende-se que se escreva um processador em GAWK que:

- Determine a lista dos domínios e das relações usadas;
- Mostre os triplos expandidos correspondentes (um triplo por linha);
- Mostre a informação contidas nos triplos, agrupadas pelo termo1 (formato Thesaurus ISO);

2.2 Descrição do Problema

Todo este problema se baseia em interpretação e filtragem de texto, sendo este encontrado nos ficheiros *.mdic* referenciados anteriormente. Há seis ficheiros diferentes que contêm informação sobre relacionamentos de diferentes temas, ou seja, diferentes domínios. Desta maneira, sabe-se que numa linha de qualquer ficheiro *.mdic* pode ser encontrado um dominio, relações normais e relações inversas e os relacionamentos em si, que envolve um ou mais termos, separados por "**:**". Decidimos resolver este problema apresentando os resultados em *HTML*.

2.3 Resolução do Problema

2.3.1 Alínea A)

Nesta primeira alínea era pedida a listagem de todos os domínios e relações existentes em um ou mais ficheiros. Assim, em cada ficheiro *.mdic*, o domínio é identificado por (**%dom: domínio**), a relação inversa por (**%inv: relação : relação**) e a relação por (**%THE: relação**), de maneira a ser possível resolver várias alíneas, alteramos o valor de **FS** para **FS=":"**, permitindo que os diferentes termos estejam repartidos pelo caratér **:**.

%THE pode identificar também uma classe que representa um conjunto de relações existente entre o primeiro e todos os outros termos apresentados à sua direita. Sendo assim, o domínio, a relação inversa e a relação são representados, por exemplo:

- **%dom($1) : corpo humano($2)**
- **%inv($1) : syn($2) : syn($3)**
- **%THE($1) : has($2)**
- **%THE<estabelecimento($1):tem_trabalhador<profissão($2):lida_com($3)**

Para o caso de o ficheiro possuir vários domínios, foi criado um array para guardar o conteúdo de cada um, correspondendo ao ($2). De maneira a apresentar os requisitos, foi usada a função match das seguintes maneiras:

```
\item match(\$1, /^%dom/);
\item match(\$1, /^%inv/);
\item match(\$1, /^%THE/);
\item match(\$1, /<(.*)/,n);
```

A expressão regular ^ *%dom* permite descobrir se uma linha começa por *%dom* que representa que nessa linha está presente o dominio. Tal como ^ *%dom*, ^ *%THE* e ^ *%inv* permitem descobrir se na linha existe uma relação ou uma relação inversa, respetivamente. Já a expressão regular utilizada no último match *<(.*)*, permite com que a função match guarde no array n, a captura de $1 que possui um simbolo < seguido de vários caracteres. Este último, é utilizado para descobrir se o relação descoberta é uma classe ou um simples relacionamento.

Quando um dominio é descoberto, esse é impresso num ficheiro *html* que tem o seu nome. Todas as relações descobertas pertencentes a esse dominio, são também escritas no ficheiro *html* correspondente.

2.3.2 Alínea B)

Em sequência do que foi pedido na alínea anterior, agora é suposto organizar os relacionamentos entre os termos através de triplos **(termo1, relacionamento, termo2)**. Assim, o problema teve que ser abordado de maneiras diferentes para

as diferentes relações, pois estas podem ser compostas por elementos (todos da mesma coluna) que são instâncias de uma *classe*, identificadas por <**classe**.

Ou seja, para solucionar este problema optámos por guardar a relação encontrada com o *match($1, /^ %THE/)*, numa variável *relacao* que posteriormente, é aplicada a função **split(relacao, ok, ¨¨**) que permite guardar no array ok todas as relações existentes numa classe. Caso *relacao* seja apenas uma relação, a função split em nada afeta.

Para responder a esta alínea utilizou-se:

- (match($1,/^ [aA-zZ]/);

A expressão regular *^ [aA-zZ]* combinada com a função match permite saber se a linha começa por qualquer letra do alfabeto, caso isso aconteça, estamos perante uma linha que diz respeito ao relacionamento. Quando essa descoberta acontece é utilizada a função *split($i, a, ")* que combinada com um ciclo, verifica se existem tuplos em $i. Se existirem, é utilizado outro ciclo para criar os triplos em si. Para tal é utilizado um algoritmo que coordena os acessos aos arrays *ok* e *a* de forma a que os triplos sejam criados de maneira correta.

2.3.3 Alínea C)

Nesta alínea era pedido que fosse construido um conjunto de páginas em HTML, onde haverá uma página por cada primeiro termo e os segundos termos hiperligam para as respetivas páginas. Como foi decidido apresentar as três alíneas em formato HTML, segue na presente alínea a explicação da passagem para a linguagem de hiperligações de todo o projeto. Para começar, visto que há uma quantidade considerável de primeiros termos e o objetivo era apresentar um por página, como iriam ser criados muitos ficheiros, decidiu-se criar uma pasta **html** para onde vão ser direcionados todos os ficheiros criados ao longo da execução do código. Assim, para construir ficheiros HTML decentes começamos por imprimir, em cada um, a estrutura básica desta markup language

```
<html>
        <head>
            <meta charset='UTF-8'/>
        </head>
    <body>
        .
        .
        .
    <\body>
        <\html>
```

A página inicial de cada ficheiro apresenta as relações, relações inversas e o domínio. O endereço da primeira página corresponde ao nome do domínio, por exemplo, para o ficheiro **vestuário.mdic**, esta denomina-se por **vestuário.html**. As hiperligações existentes nessa página situam-se no nome das relações (não

no das relações inversas), onde cada página que segue a hiperligação escolhida é endereçada pelo domínio e a relação. A título de exemplo, após carregar na relação **nt** do domínio acima, a página seguinte denomina-se **vestuario%20nt%20.html**. Nesta página encontramos a lista dos primeiros termos de cada triplo associado a essa relação, cada um correspondendo a hiperligação que nos direciona para a página que contém os triplos correspondentes. Em seguimento da denominação das páginas, a página que contém os triplos tem como endereço **vestuarioroupa%20nt%20.html**. Em cada página á dada a opção de voltar para a anterior. Em anexo seguem as imagens dos exemplos aqui referidos.

3 Processador de textos preanotados com Freeling

3.1 Enunciado 2.2

Em **natura.di.uminho.pt/ jj/pl-18/TP1/CORPORA2/** podemos encontrar os ficheiros, que foram previamente anotados com Freeling (ferramenta de análise linguística). Após serem analisados os extratos é pedido:

- contar o número de extratos
- calcular a lista dos personagens do Harry Potter (nome próprio) e respetivas ocorrências
- calcule a lista dos verbos, substantivos, adjectivos e advérbios PT: e crie um ficheiro com cada uma destas listas
- determinar o dicionário implícito no corpora – lista contendo os lema, pos e palavras dele derivadas.

3.2 Descrição do Problema

Como foi dito anteriormente, para este problema serão utilizados cinco ficheiros que contêm blocos de texto e informação morfossintática de palavras denominados extratos. Assim, em cada ficheiro encontramos pelo menos um extrato. Estes estão separados por linhas em branco e em cada um existem sete colunas com diferentes informações sobre palavras, sendo as que se destacam:

- número de linha - **$1**
- palavra - **$2**
- lema - **$3**
- pos-tag - **$4**
- pos - **$5**

3.3 Resolução do problema

3.3.1 Alínea A)

Nesta alínea era pedido para calcular o número total de extratos de um ficheiro. Sabendo que os mesmo estão separados por linhas em branco, foi decidido contar o número de linhas brancas. No entanto, o total do número destas linhas seria igual ao número de extratos -1. Inicialmente pensou-se em iniciar uma variável a 1 e incrementar sempre que encontrava uma linha branca, porém, ao compilar vários ficheiros haveria uma descompensação, dando um resultado menor do que o suposto. Assim, foi decidido acrescentar em cada ficheiro uma linha branca no início. Desta maneira, as linhas foram encontradas através da expressão:

```
if(!NF)
    ext++;
```

Onde NF é o number of fields, ou seja, sabemos que é uma linha branca quando não existe nenhum campo na mesma, e incrementamos a variável **ext**. No fim, é imprimido este valor.

3.3.2 Alínea B)

Nesta alínea era pedido para construir a lista com todas as personagens dos ficheiros **harrypotter** e respetivas ocorrências. Sabemos que uma fração de texto é uma personagem através da sua pos-tag ($5), quando esta for um nome próprio, neste caso identificado por **NP**, sabemos que estamos na presença de uma personagem. Em termos de código, foi usada a função **match** com a expressão regular

```
/^NP/
```

De maneira a calcular o número de occorrências de cada nome ($2), foi criado um array que guarda as mesmas.

```
if(match(\$5,/\^NP/)){
    nomes[\$2]++;
```

No fim, é usado um ciclo para percorrer o **nomes** e imprimir os requisitos.

3.3.3 Alínea C)

Nesta alínea, era pedido que construíssemos diferentes ficheiros que contenham adjetivos, advérbios, substantivos e verbos. O método utilizado foi semelhante ao anterior, no entanto, foi necessária a criação de um array para cada ficheiro. Foram usadas as seguintes funções para encontrar o pretendido:

```
if(match($5,/^R/))
    adv[$2]++;
```

```
if(match($5,/^A/))
     adj[$2]++;

if(match($5,/^N/))
     subs[$2]++;

if(match($5,/^V/))
     verb[$2]++;
```

Onde cada letra das expressões regulares identifica, respetivamente, advérbios, adjetivos, substantivos e verbos. No final, é usado um ciclo por cada array criado de maneira a percorrer e imprimir o conteúdo de cada um para o respetivo ficheiro.

3.3.4 Alínea D)

Por último, foi requisitado que se construisse um dicionário onde eram identificados os lemas e respetivas palavras e pos. Como, para cada lema, podem existir várias palavras foi criado uma estrutura diferente, sendo esta um array de arrays de palavras identificadas pelo respetivo lema e um array que identifica, para cada lema, a respetiva pos.

```
    lemas[$3][$2];
    pos[$3] = $5;
```

No final, é usado um ciclo para a impressão dos requisitos no respetivo ficheiro.

4 Conclusão

Depois de terminado o primeiro trabalho prático da unidade curricular de Processamento de Linguagens, podemos concluir que a realização do mesmo foi muito importante, no sentido em que nos ajudou bastante a consolidar a matéria leccionada nas aulas da disciplina, desde o desenvolvimento de Expressões Regulares (ERs) para filtrar e transformar textos até ao uso da linguagem **GAWK** para desenvolver as soluções necessárias para resolver os enunciados. Este trabalho permitiu também aumentar os nossos conhecimentos de *HTML* visto ser a primeira vez que lidamos com a linguagem.

No geral podemos dizer que estamos satisfeitos com o trabalho desenvolvido, tendo este, sido bastante útil para o nosso crescimento como futuros engenheiros informáticos.

5 Anexos

Dominio -> vestuario

Relação inversa ->ntbt

Relação inversa ->domvoc

Relação inversa ->haspof

Relação inversa ->instiof

Relação -> nt

Relação -> inst

Relação -> has

Relação -> EN

Figura 1: Página Principal

Back

nt

vestuário

roupa

Figura 2: Página Relação - nt

Back

(roupa, nt, roupa feminina)

(roupa, nt, roupa de cama)

Página Triplos -> termo1 = roupa