Processador de Thesaurus

José Carlos Lima Martins A78821 and Miguel Miranda Quaresma A77049

University of Minho, Department of Informatics, Braga, Portugal
e-mail: {a78821,a77049}@alunos.uminho.pt

Resumo Este documento serve de apoio ao desenvolvimento de um Processador Thesaurus explicando todas as decisões tomadas de modo a obter o mesmo. Inicialmente será explicada a estrutura dos ficheiros de entrada (extensão .mdic) bem como mostradas as decisões tomadas de modo a obter o produto final. Por fim, serão apresentados exemplos de utilização do mesmo.

1 Introdução

O presente projeto (Processador de Thesaurus) visa ser um sistema de processamento de dicionários Thesaurus que recorre a expressões regulares (ERs) para filtrar e transformar os mesmos, extraindo e tratando a informação mais relevante de forma eficiente. Para atingir este objetivo é usada a linguagem de filtragem e tratamento de dados `AWK` visto ser uma **DSL**(Domain Specific Language) com foco em dados semi-estruturados.

2 Preliminares

De modo a compreender melhor o desenvolvimento deste projeto é importante saber e compreender a estrutura dos dicionários (ficheiros .mdic). A estrutura de um dicionário pode ser dividido em três conceitos:

- linhas começadas por '#': comentários a ignorar
- diretivas gerais:
  - linhas começadas por '%dom: dominio': indica que até ao aparecimento de nova linha começada por '%dom:' todos os termos são de 'dominio', sendo dom uma relação e a sua inversa é voc
  - linhas começadas por '%inv: relação1 : relação2': indica que a relação1 tem como inversa a relação2
- linhas começadas por '%THE': indicam tabelas de relações, com as seguintes caracteristicas:
  - a linha inicial possui as relações bem como as classes da coluna correspondente
  - cada linha tem 1 ou mais termos, menos a inicial que possui apenas relações e classes
  - os termos são separados por ':' daqueles com que se relacionam
  - a relação entre o termo da coluna 1 e da coluna N é dado pela relação N da linha inicial

- na presença de vários termos com a mesma relação, podem ser agrupados por '|'
- quando uma relação na linha inicial possui <classe, ou seja 'relação<classe', significa que o elementos dessa coluna são instancia da classe
- quando '%THE<classe' significa que o termo 1 é instancia da classe

Pode-se finalmente, passar ao desenvolvimento do processador Thesaurus.

3 Desenvolvimento

3.1 Exe 1

O processamento iniciou-se pela identificação dos domínios e relações presentes no dicionário. Para tal, e tendo em conta a estrutura dos documentos, foi usado como `Field Separator` o carácter ':'. Tendo a conta a sintaxe de Padrão → Ação inerente ao `AWK` foram considerados relevantes os seguintes padrões:

1. `/^%dom/`: padrão que identifica novo domínio no início de uma linha
2. `/^%THE/`: padrão que identifica início de tabela de relações indicadas na presente linha

No caso do padrão `/^%dom/` a ação compreende a remoção de espaços que precedam o domínio e o armazenamento do mesmo numa matriz `ind`.

```
/^%dom/     {sub(/^ /,"",$2); ind[0][$2]++}
```

Quando é encontrado o padrão `/^%THE/` são percorridos todos os campos (**i.e.**relações) e quando um dada relação não vazia (`$j!=" " && $j!=""`) é encontrada são-lhe removidos os espaços precedentes e subsequentes removendo também, eventualmente, a indicação de classe. De seguinta estaes são também armazenados na matriz `ind`.

```
/^%THE/     {for(j=2;j<=NF;j++){
                if($j != " " && $j!=""){
                    sub(/^ /,"",$j);
                    sub/(\s)|(<(.*)?)/,"",$j);
                    !ind[1][$j]++;
                }
             }
            }
```

Por fim, quando o final de ficheiro é encontrado (`EOF`) são impressos os domínios e as relações recorrendo, para isso, às funções `printDominios` e `printRelations` respetivamente.

Usage

4 Exe 2

O objetivo deste exercício é mostrar os triplos expandidos(um por cada linha) correspondentes. Como tal, há a necessidade de percorrer os ficheiros por completo. Consideramos como Field Separator ':'. Consoante o início da linha, o processamento é indicado de seguida:

- `/^%dom/`: guarda-se o dominio indicado numa variável até aparecer uma nova linha com o mesmo padrão altura em que o dominio é atualizado
- `/^%inv/`: guarda num índice `inv` a relação e a sua respetiva inversa
- `/^%THE/`: as relações são armazenadas num índice `relation` e as classes (quando aplicável/existentes) num índice `class`
- `$1 !~/^%/ && $0 !~ /^#/ && $0 != ""` (**i.e.**tabelas de relações): os triplos são armazenados no índice `triples` da seguinte maneira e consoante as seguintes condições:
  - termo1 relacionado com o dominio(**dom**) bem como a respetiva inversa(**voc**)
  - caso exista uma classe associada ao '%THE' é adicionado os triplos correspondentes ao **instance of**(iof) e **tem como instancia**(inst) envolvendo o termo1 e a classe
  - são percorridos o resto dos campos (termos da lado direito), separando-os por | usando o `split` obtendo as seguintes triplos dependendo do caso:
    - os triplos que relacionam o termo com o dominio (dom e voc)
    - o triplo envolvendo o termo1, a relação da coluna correspondente e o termo, sendo que caso a relação possua inversa(verificado no indice inv) é adicionado o triplo
    - caso a coluna possua classe, o termo é relacionado com a mesma(iof e inst)

Após a adição de todos os triplos no indice triples, ou seja após percorridos todos os ficheiros(END) é percorrido esse mesmo indice imprimindo cada um. Este indice possui como elemento diferenciador o triplo e como conteudo se ocorreu, de modo a retirar triplos repetidos. É também bom referir que todos os elementos, seja classes, relações ou termos é lhes retirado espaços que estejam a mais antes e após o mesmo.

Usage O exemplo mais dificil de utilização deste programa é:

```
gawk -f exe2.awk *.mdic
```

Contudo pode ser também interessante guardar o resultado num ficheiro:

```
gawk -f exe2.awk *.mdic >> file.txt
```

4.1 Exe 3

Neste ultimo exercicio, aproveitamos o resultado obtido no anterior, ou seja os triplos obtidos anteriormente são usados como input para este. Sendo assim, a todas as linhas retiramos os parenteses, dividindo de seguida pelas virgulas, removendo depois ao termo1 e ao termo2 os espaços excedentes, isto é, que estejam antes ou após o conteúdo. O termo1 e o termo2 são de seguida, guardado num indice triples separados apenas por uma virgula (`triples[i]=triple[1] "," triple[3];`). É também alterado o valor num indice ind de modo a sabermos no final se será criado ou nao o ficheiro correspondente a cada termo de modo a sabermos futuramente se pode haver uma hiperligação para o mesmo. No fim de percorridos todo o input(END), percorremos todo o indice triples,em que para cada elemento tem-se em atenção que se for a primeira vez a ser criado o ficheiro adicionamos o encoding e a macro de lista, isto é:

```
if(create[aux[1]]==""){
    print "<meta charset=\"utf-8\">" >> aux[1] ".html";
    print "<ul>" >> aux[1] ".html";
    !create[aux[1]]++;
}
```

Para além disso, antes de adicionar ao mesmo o hyperlink verifica-se se o podemos fazer verificando o indice ind. Em caso negativo é colocado apenas colocado o nome do termo2, como se pode verificar em:

```
if(ind[aux[2]]!="")
    print "<li><a href=\"" aux[2] ".html\">" aux[2] "</a></li>" >> aux[1] ".html"
else print "<li>" aux[2] "</li>" >> aux[1] ".html"
```

Após percorrido o indice triples, é altura de adicionar a macro de fecho de lista (¡/ul¿) a todos os ficheiros e, como pelo indice create sabe-se todos os ficheiros criados basta percorrê-lo adicionando a macro em falta, como demonstrado de seguida:

```
for(y in create){
    print "</ul>" >> y ".html";
}
```

Usage Como anteriormente referido foi usado o output do exercicio anterior e como tal de modo a obter os resultados esperados o programa deve ser executado preferencialmente usando uma das seguintes maneiras:

- `gawk -f exe2.awk *.mdic | gawk -f exe3.awk`

- ```
  gawk -f exe2.awk *.mdic >> file.txt
  gawk -f exe3.awk file.txt
  ```

5 Conclusão
  ```