

Universidade do Minho
Licenciatura em Engenharia Informática
Processamento de Linguagens



Processamento da Wikipedia

Ano Lectivo de 2010/2011

54745 **André Pimenta**
54738 **João Gomes**
54802 **Milton Nunes**

3 de Abril de 2011

Resumo

Este projecto consiste sobretudo na utilização de expressões regulares. Pretende-se, através do uso do Special Export, que sejam exportadas páginas do wikipedia para que depois através da identificação de determinados padrões em XML, se possam definir expressões regulares que extraiam os dados que pretendemos, isto com o auxílio das ferramentas Flex.

Através da linguagem C são tratados os dados e criado um output, em HTML, com os resultados obtidos.

Neste relatório daremos sobretudo importância aos padrões de XML encontrados, às expressões regulares que desenvolvemos e também ao output do trabalho, ou seja ao nosso produto final, para além de um diagrama de estados referente a todas as expressões importantes.

Conteúdo

Conteúdo	1
1 Contextualização	2
1.1 Estrutura	2
1.2 Motivação e objectivos	2
1.3 Introdução	2
2 Desenvolvimento do processador da Wikipedia	3
2.1 Expressões Regulares	3
2.2 Expressões Regulares e estados correspondentes	4
2.3 Estrutura de Dados	6
2.4 Makefile	6
2.5 Diagrama de Estados	7
2.6 Testes estatísticos	8
2.7 Output do programa	9
3 Conclusão	11
3.1 Elementos do Grupo	12

Capítulo 1

Contextualização

1.1 Estrutura

O presente relatório é constituído por 3 capítulos: **Contextualização** onde é feita uma breve contextualização do problema dentro da engenharia de processamento de linguagens, **Desenvolvimento do processador da Wikipedia** onde são descritos todos os passos do desenvolvimento do processador da wikipedia e **Conclusão** onde é feita uma breve conclusão sobre o trabalho realizado.

1.2 Motivação e objectivos

O desenvolvimento deste trabalho implica o uso de expressões regulares e o uso da ferramenta Flex. As expressões regulares, como foi leccionado nas aulas de Processamento de Linguagens, permite-nos identificar sequências, palavras ou padrões de caracteres em ficheiros.

Isto torna-se muito útil pois permite-nos retirar de um ficheiro determinados padrões para usarmos posteriormente no nosso software, que é o que se pretende neste trabalho, uma selecção de determinados padrões para depois usar como output.

Para isso usa-se também o Flex, que consiste numa ferramenta que vai identificar no ficheiro XML os padrões definidos por nós através de regras definidas através de expressões regulares.

É portanto, nosso objectivo aplicar as expressões regulares e o uso da ferramenta Flex, desenvolvendo as suas potencialidades e adquirindo o máximo de conhecimento acerca destas.

1.3 Introdução

Pretende-se com este trabalho a aplicação dos conhecimentos adquiridos nas primeiras aulas da disciplina de Processamento de Linguagens. O trabalho consiste em criar um filtro, através da aplicação de expressões regulares, para estruturar um conjunto de dados extraídos de páginas da Wikipedia, em formato XML.

O output deste filtro, que foi obtido através do Flex, será gerado em páginas HTML contendo as informações extraídas, depois de estas serem tratadas através de algumas funções desenvolvidas na linguagem C.

Capítulo 2

Desenvolvimento do processador da Wikipedia

2.1 Expressões Regulares

Para criar as expressões regulares é importante que antes sejam identificados os padrões em XML que representem os conteúdos que se pretendem retirar do texto a processar.

Portanto foi preciso analisar os ficheiros em XML extraídos do Wikipedia e concluímos que os seguintes padrões definem os conteúdos que pretendemos:

```
<title>Train</title> -> titulo
<timestamp>2011-03-16T17:59:48Z</timestamp> -> ultima revisão
<username>Fæ</username> -> revisor
==Types of trains== -> secção
[[rail transport]] -> links internos
www.* -> links externos
```

De referir que por links externos consideramos todos os links iniciados por 'www.' e represente dominios validos, e não apenas os que se encontram na secção de links externos das respectivas paginas do wikipedia.

Nas secções consideramos também apenas as secções e não as subsecções.

Encontrados os padrões que indicam os conteúdos definidos no enunciado do trabalho desenvolvemos as expressões regulares, e através do uso do egrep confirmamos se os resultados eram os pretendidos.

```
titulo = egrep -o '<title>.*</title>' Train.xml
data da ultima revisão = egrep -o '<timestamp>.*</timestamp>' Train.xml
revisor = egrep -o '<username>.*</username>' Train.xml
sectios e subsections = egrep -o '^=[}{][^=]+=$' train.xml --color=auto
links internos = egrep -o '[[]{2}[a-zA-Z0-9_]]+[[]{2}' train.xml --color=auto
links externos = egrep -o '[w]{3}\.'
```

2.2 Expressões Regulares e estados correspondentes

Com as expressões apresentadas acima conseguimos filtrar toda a informação que pretendemos, no entanto esta ainda tem de ser tratada e em alguns casos (como o da revisão) para facilitar e melhorar o uso destas mesmas expressões usamos estados no flex.

Definimos então os seguintes estados:

- page - sempre que que é encontrada a etiqueta <page> entramos dentro do estado page onde se poderá encontrar toda a informação que pretendemos obter das respectivas paginas do Wikipedia.

```

25 "<page>"      BEGIN(page);
26 <page>{
27
28     "<title>"      {BEGIN(title_sc);}
29     "<revision>"    BEGIN(revision);
30     {anything}      {;}
31     "</page>"      BEGIN(INITIAL);
32 }

```

- title - sempre que que é encontrada a etiqueta <title> e nos encontramos dentro do estado page entramos dentro do estado title onde se encontra o titulo da pagina wikipedia a ser processado.

```

33 <title_sc>{
34     .*.*          BEGIN(page);
35     [^<]+         {doc = initDocument();doc = insertTitulo(yytext,doc);lista = guardaDoc(doc,lista);}
36     "</title>"      BEGIN(page);
37 }

```

- revisão - sempre que é encontrada a etiqueta <revision> passaremos então para o estado revisão onde se encontra a informação relativa á revisão de documento e onde poderemos passar para o estado autor ou estado data de revsão.
Dentro deste quando se encontrar a etiqueta </revision> abandonará-se este estado e passa-se para o estado que precede este, o estado page.

```

38 <revision>{
39     "<timestamp>"    BEGIN(time);
40     "<username>"      BEGIN(author);
41     ^[=]{2}        BEGIN(section);
42     [w]{3}\.       BEGIN(ext);
43     [[]]{2}        BEGIN(intern);
44     {anything}      {;} /* do nothing*/
45     "</revision>"    BEGIN(page);
46 }

```

- autor - este estado pode ser encontrado sempre que se encontrar a equita <username> dentro do estado revisao.
Dentro deste estado encontre-se o nome do autor da ultima revisão até se encontrar a etiqueta </author> que faz com que passemos ao estado anterior revisão.
- data de revisão - entramos neste estado sempre que encontramos a etiqueta <timestamp> e estamos dentro do estado revision, dentro deste estado temos acesso á data da ultima

```

47 <author>{
48     [^<]+           {doc = insertAutorLastRec(yytext, doc);}
49     "</username>"    BEGIN(revision);
50 }

```

revisão feita no documento.

Este estado é abandonado quando se encontra a etiqueta `</timestamp>` e passamos para o estado anterior.

```

51 <time>{
52     [^<]+           {doc = insertDateLastRec(yytext, doc);}
53     "</timestamp>"    BEGIN(revision);
54 }

```

- ligação interna - estando dentro do estado revision sempre que se encontra a expressão "[["o estado de ligação interna é activado.
Dentro deste encontra-se uma ligação interna até voltar a encontrar "]"ou ' | ' (este é encontrado sempre que existem ligações internas com o mesmo nome mas que podem ser identificadas com nomes diferentes.)

```

63 <intern>{
64     [^:|+/"|"]       {doc = insertLinkInter(yytext, doc);}
65     [^:|+|/|]         {doc = insertLinkInter(yytext, doc);}
66     {anything}        {BEGIN(revision);}
67 }

```

- ligação externa - as ligações externas encontram-se sempre que se encontra a expressão "www.", após encontrar-mos este padrão entramos dentro do estado ligação externa. Dentro do estado de ligação externa podemos encontrar todas as referencias a ligações externas da pagina Wikipedia.

```

59 <ext>{
60     [a-zA-Z0-9_\.]+\.[a-zA-Z]+ {doc = insertLinkExt(yytext, doc);}
61     {anything}                 BEGIN(revision);
62 }

```

- secção - sempre que se encontrar a expressão "=="entraremos no estado de secção dentro desta encontram-se todas as secções da pagina Wikipedia até se encontrar a expressão "=="que faz com que voltemos ao estado revision.
Devemos dar atenção neste estado que podemos estar perante subsections que são identificados com a expressão "===".

```

55 <section>{
56     [^=]+           {doc = insertLinkSec(yytext, doc);}
57     [=]+           BEGIN(revision);
58 }

```

2.3 Estrutura de Dados

Depois de definidas as expressões regulares e utilizando o Flex foram extraídos os conteúdos pretendidos. Para que esses dados pudessem ser tratados foi necessário criar uma estrutura de dados onde eles são guardados. Definimos então a seguinte estrutura de dados:

```
typedef struct links{
char* nomeLinks;
struct links *next;
} *Links;

typedef struct Document{
char* titulo;
char* autorLastRev;
char* dateLastRev;
int nInter;
Links linkInter;
int nExt;
Links linkExt;
int nSec;
Links linkSec;
} *Docs, DOC;

typedef struct listDocs{
Docs doc;
struct listDocs *next;
} *List;
```

Esta estrutura permite-nos guardar os dados extraídos, isto é, permite nos guardar o título, o autor, a data da última revisão, o número de links externos/internos e quais são, e o número de secções e os seus nomes para cada página da Wikipedia contida no ficheiro XML.

Desta forma conseguimos tratar os dados(e.g ordenar todo alfabeticamente) para que depois conseguíssemos gerar o output como pretendemos.

2.4 Makefile

Apresentamos a Makefile que por nós desenvolvida e que vêm permitir a instalação do programa de uma forma simples e correcta.

A criação desta Makefile é importante pois vêm juntar varios comandos que seriam necessários executar para que se corresse o programa. Desta forma com o simples comando `make install` temos o programa pronto a executar.


```

1 LEX=scan
2 NAME=processar_Wiki
3 objects=TrabInputs.o htmlCreate.o
4 installFiles = estrutura.h htmlCreate.c htmlCreate.h processar_Wiki.c scan.l TrabInputs.c TrabInputs.h
5 flags=-c
6
7 $(NAME).c:$(NAME).c $(objects)
8 cc -o $(NAME) $(NAME).c $(objects) #-lfl
9 $(NAME).c:$(LEX).l
10 flex -o $(NAME).c $(LEX).l
11 TrabInputs.o:TrabInputs.c TrabInputs.h estrutura.h
12 gcc $(flags) TrabInputs.c
13 htmlCreate.o:htmlCreate.c htmlCreate.h estrutura.h
14 gcc $(flags) htmlCreate.c
15 clean:
16 rm $(objects) $(NAME).c
17 teste:$(NAME)
18 ./$(NAME) < teste.xml
19
20 install:$(NAME)
21 mkdir -p Install
22 mkdir -p XML
23 mv $(installFiles) ./Install/
24 cp Makefile ./Install/
25 rm $(objects)

```

Figura 2.1: Makefile

2.5 Diagrama de Estados

Desenvolvemos também o nosso diagrama de estados, como ilustra a imagem abaixo. Esta representa os estados possíveis e o que leva a uma alteração dos mesmos.

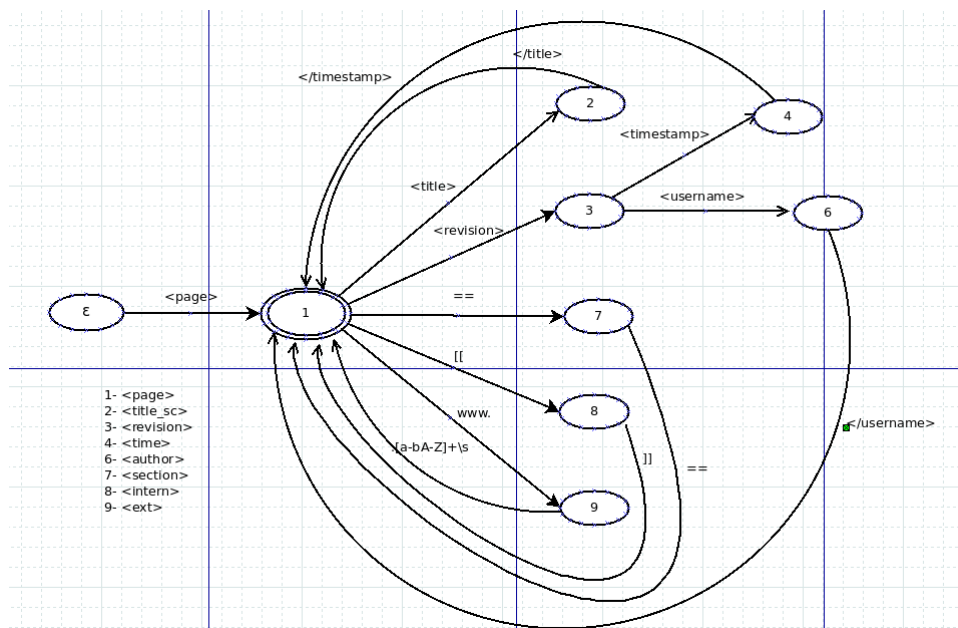


Figura 2.2: Diagrama de Estados

No nosso diagrama de estados cada nodo identifica o estado em que se encontra. E cada aresta contém a expressão regular que leva à transição de estado para estado.

2.6 Testes estatísticos

Concluída a aplicação decidimos realizar testes estatísticos. Estes testes pretendem verificar o tempo que demora a executar ficheiros de tamanhos diferentes.

Realizados os testes, aqui estão os resultados obtidos (pela seguinte ordem: Tamanho(MB), Tempo real(s) e N° titulos do ficheiro):

2,4mb — 0,4 seg — 100 artigos
9,5 mb — 0,8 seg — 500 artigos
17,5 mb — 1,1 seg — 1000 artigos
51,3 mb — 1,5 seg — 5000 artigos
90,8 mb — 2,3 seg — 10000 artigos
313,4 mb — 3,1 seg — 50000 artigos

Estes ficheiros foram criados através do ficheiro original da Wikipedia em português de cerca de 3.6Gb usando a ferramenta felx para criar ficheiros com a quantidade de artigos que desejamos.

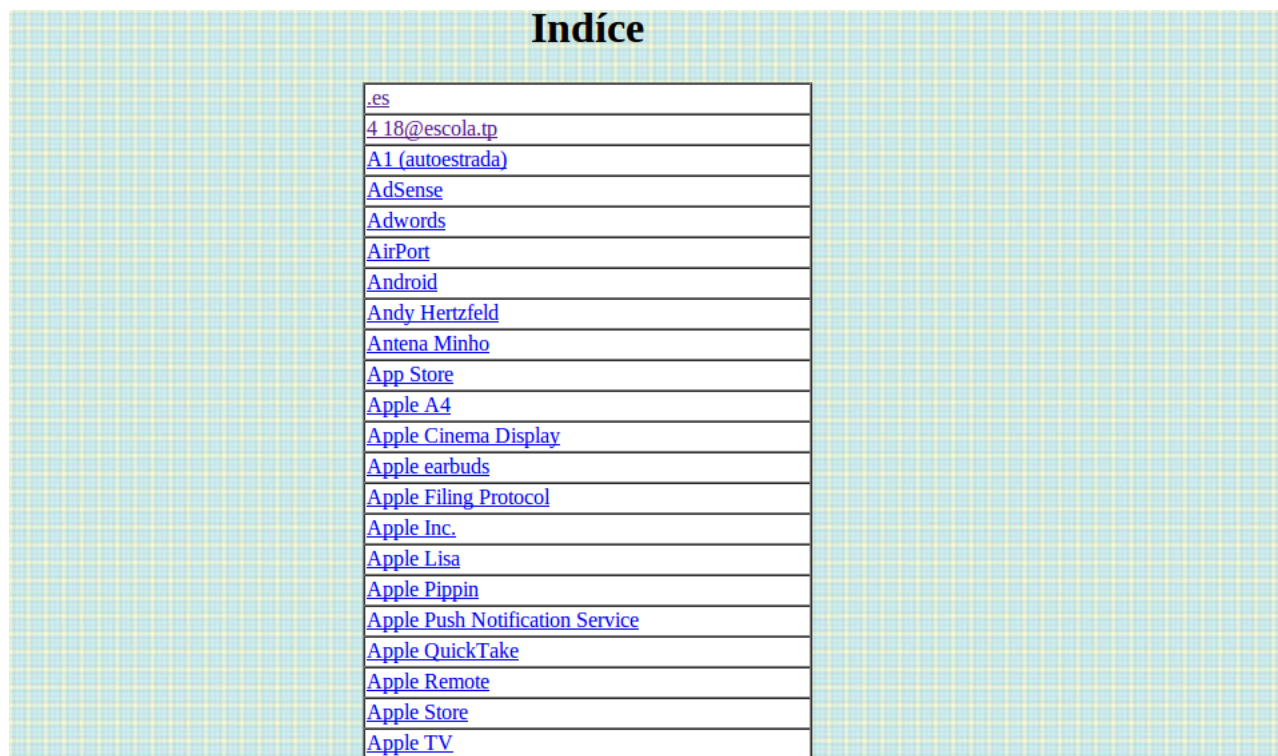
```
13 %%  
14 FILE *f = fopen("500.xml", "w+");  
15 long total=0;  
16 char fim[]="</mediawiki>";  
17  
18 "<page>" {total++;if(total>=500) {fprintf(f,fim,yytext);printf("acabei :-\n"); fclose(f); return 1;} else fprintf(f,"%s",yytext);}  
19  
20  
21 {anything} {fprintf(f,"%s",yytext);}  
22 <<EOF>> {return 1;}  
23
```

Figura 2.3: Flex para criação de ficheiros

2.7 Output do programa

Concluída a parte referente às expressões regulares e ao Flex, ou seja, assegurados os dados pretendidos no enunciado, a informação foi tratada, através de funções desenvolvidas na linguagem C, para posteriormente surgir no output do programa.

Esse output, tal como pedido no projecto, foi desenvolvido em HTML, e para cada ficheiro da wikipedia processado é gerado um índice que contém todas as páginas existentes no ficheiro XML a processar. Através deste índice podes aceder a todas as páginas e consultar os dados destas. Apresentamos em baixo imagens que ilustram o outPut gerado pelo nosso programa.



.es
4 18@escola.tp
A1 (autoestrada)
AdSense
Adwords
AirPort
Android
Andy Hertzfeld
Antena Minho
App Store
Apple A4
Apple Cinema Display
Apple earbuds
Apple Filing Protocol
Apple Inc.
Apple Lisa
Apple Pippin
Apple Push Notification Service
Apple QuickTake
Apple Remote
Apple Store
Apple TV

Figura 2.4: Índice

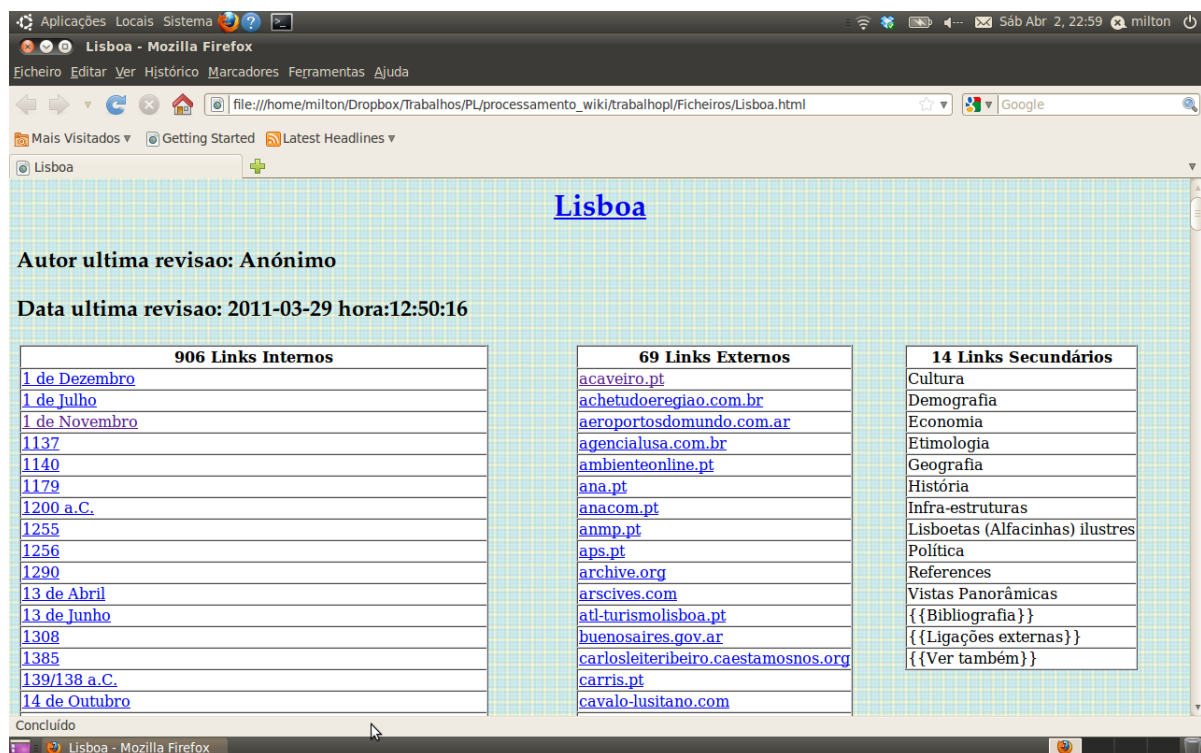


Figura 2.5: Output do ficheiro lisboa.xml

Para gerar este output é necessário correr o programa, utilizando os seguintes comandos: `make install` e `./processar_Wiki nome_ficheiro.xml`

O output inclui todos os dados que o enunciado pretendia que retirássemos dos ficheiros XML, isto é, o título, autor da última revisão, data da última revisão, links externos, internos e secções. Estas últimas 3 encontram-se nas tabelas, como ilustra a imagem acima.

Capítulo 3

Conclusão

Este trabalho, apesar de ser um exemplo simples, ajudou-nos a perceber a potencialidade que as expressões regulares têm, isto é, o que nos permite fazer de forma simplificada.

As expressões regulares, revelam-se portanto, extremamente úteis para encontrar informação em ficheiros. Sendo que podemos estar a falar de milhares de ficheiros, com milhares de caracteres cada um, facilmente se percebe que a utilidade que oferece o uso das expressões regulares.

3.1 Elementos do Grupo



Figura 3.1: André Pimenta



Figura 3.2: João Miguel



Figura 3.3: Milton Nunes