

Processamento de Linguagens (3º Ano MIEI)

Trabalho Prático 1b - FLEX

Relatório de Desenvolvimento

Fábio Luís Baião da Silva
(A75662)

João da Cunha Coelho
(A74859)

Luís Miguel Moreira Fernandes
(A74748)

22 de Abril de 2017

Resumo

O Flex foi criado em 1987 pelo professor de Ciências da Computação da Universidade da Califórnia, Vern Edward Paxson e surge como alternativa grátis e *open-source* ao Lex. É um programa que gera analisadores léxicos (também conhecidos como *scanners*) com inúmeras utilidades para o dia-a-dia da população. É frequentemente usado junto do gerador de *parser yacc* nos sistemas operativos *BSD-derived*.

Conteúdo

1	Introdução	2
2	Processador de Named Entities	3
2.1	Estrutura do Ficheiro XML	3
3	Apresentação da Solução	4
3.1	Análise do problema	4
3.2	Outras Cenas	4
4	Apresentação das Queries e Resoluções - TEMA1	5
4.1	Queries de Resolução Obrigatória	5
4.2	Queries Adicionais	5
5	Apresentação de Resultados	6
5.1	Execução	6
5.2	Exemplos	6
6	Processador de inglês corrente	7
7	Apresentação das Queries e Resoluções - TEMA 2	10
7.1	Queries de Resolução Obrigatória	10
7.2	Queries Adicionais	10
8	Apresentação de Resultados	11
8.1	Execução	11
8.2	Exemplos	11
9	Conclusão	13

Capítulo 1

Introdução

O primeiro contacto com o gerador Flex despertou grande curiosidade, uma vez que consiste em filtros a aplicar a textos que tornam o reconhecimento e análise mais fácil e intuitiva. O uso da linguagem C ao gerar filtros em Flex é, também, de grande utilidade, sendo que é uma linguagem já conhecida e trabalhada ao longo do curso.

Quanto ao tema, a escolha recaiu sobre o tema **2.4. Processador de Named Entities**, devido à clareza do enunciado e à dimensão das dificuldades que este nos poderia trazer e que gostaríamos de conseguir superar, no entanto também desenvolvemos uma solução para o tema **2.1 Processador de Inglês Corrente**, que também merecerá uma secção própria neste relatório.

Capítulo 2

Processador de Named Entities

2.1 Estrutura do Ficheiro XML

De seguida, é sucintamente explicada a estrutura do dialeto XML chamado *ENAMEX*, em análise neste trabalho.

```
<?xml version="1.0" encoding="UTF-8"?>
<document>
<ENAMEX> Vivia </ENAMEX> este herói no <ENAMEX TYPE="CITY"> Rio de
Janeiro </ENAMEX>, <ENAMEX TYPE="COUNTRY">Brasil</ENAMEX> e era
professor não se sabe de que doutrinas no ano de
<TIMEX TYPE="DATE"> 1710 </TIMEX>, quando os franceses comandados por
<ENAMEX TYPE="PERSON"> Duclerc </ENAMEX>, atacaram a cidade.
O governador portou-se mal, e permanecia numa deplorável inacção, quando
<ENAMEX TYPE="PERSON"> Bento do Amaral </ENAMEX> à frente dos seus
estudantes e de paisanos armados, saiu a tomar o passo aos invasores,
repelindo-os energicamente, e dando lugar a que o ataque se malograsse
e os agressores ficassem prisioneiros.
<ENAMEX> Não </ENAMEX> tardou <ENAMEX TYPE="PERSON"> Dugua-Trouin </ENAMEX>
a vir tomar a desforra.
</document>
```

As etiquetas *ENAMEX* e *TIMEX* permitem identificar, respetivamente, entidades representadas pelo nome próprio ou datas. Como podemos ter cidades, pessoas ou países representados pelo nome próprio, associa-se também um tipo à etiqueta.

Capítulo 3

Apresentação da Solução

3.1 Análise do problema

Perante a necessidade de escrever um processador capaz de analisar um documento como o mencionado acima e produzir um índice HTML para cada um dos tipos da etiqueta *ENAMEX*, começou-se por definir a estrutura usada para armazenar os nomes próprios de cada tipo, durante o processamento do ficheiro. A escolha recaiu pelo uso de arrays, um por tipo, particularmente os *pointer arrays* da biblioteca **GLib**, por facilidade de utilização e por terem sido abordados em aulas práticas.

```
GPtrArray *cidades, *países, *pessoas;
```

Nota também para o uso de variáveis que contabilizam as ocorrências de cada um dos tipos *ENAMEX*, auxiliares posteriormente na impressão dos índices HTML.

Para determinar o maior e o menor ano mencionados no texto criaram-se duas variáveis, uma para cada ano, cujo valor será controlado por intermédio de comparações de valor, à medida que vamos percorrendo o ficheiro.

```
float menor = +INFINITY;  
float maior = -INFINITY;
```

```
(...)
```

```
<DATA>{NUM} { int ano = atoi(yytext);  
if (ano < menor) menor = ano;  
if (ano > maior) maior = ano; }
```

3.2 Outras Cenas

Capítulo 4

Apresentação das Queries e Resoluções - TEMA1

Neste capítulo serão explicadas as queries realizadas. Na primeira secção apresentar-se-ão as queries obrigatórias sendo, de seguida, apresentadas as queries criadas adicionalmente pelo grupo.

4.1 Queries de Resolução Obrigatória

- a) Produzir um índice em HTML com as Pessoas, Países e Cidades referidas.
- b) Usar o Google Maps para localizar as cidades referidas.

4.2 Queries Adicionais

- a) Adicional 1
- b) Adicional 2

Capítulo 5

Apresentação de Resultados

Neste capítulo serão apresentados exemplos de utilização.

5.1 Execução

5.2 Exemplos

Abrindo o menu principal aparece o seguinte resultado:

Utilizando o menu de navegação superior para clicar em Cliente, obtem-se a página relativa aos dados do cliente. A partir do menu lateral, na secção de Consultas, é possível consultar as queries realizadas.

Capítulo 6

Processador de inglês corrente

A primeira etapa da resolução deste problema foi tornar o filtro FLEX capaz de detetar as contrações de uso corrente e constante, tais como *It's*, *I'm*, *You're*, entre outras. Sem a cobertura destes casos seria difícil considerar positiva a solução desenvolvida pelo grupo, mas com expressões como as que se podem ver na figura abaixo foi possível alcançar bons resultados.

```
[Ss]han't      { fprintf(f, "%shall not", yytext[0]); }
[Ww]on't       { fprintf(f, "%scill not", yytext[0]); }
[Cc]an't       { fprintf(f, "%cannot", yytext[0]); }
n't            { fprintf(f, " not"); }

'm             { fprintf(f, " am"); }
'll            { fprintf(f, " will/shall"); }
're            { fprintf(f, " are"); }
's             { fprintf(f, " is"); }
```

Figura 6.1: Contrações mais recorrentes.

Há, porém, situações em que surge dúvida sobre a palavra contraída, uma vez que a contração pode representar dois vocábulos diferentes. Um exemplo é a contração *'ll*, que pode comprimir tanto *will* como *shall*. Em abreviaturas dúbias como esta, optou-se por moldar o processador no sentido de substituir a contração pelas várias palavras que pode representar, separadas por /, cabendo depois ao utilizador fazer o seu juízo de valor e editar, se assim o preferir, a solução dada pelo programa, algo permitido pelo HTML no qual é exposto o texto de *output*.

```
[Hh]ow'll      { fprintf(f, "%cow will", yytext[0]); }
[Hh]ow's       { fprintf(f, "%cow has/is", yytext[0]); }
'd             { fprintf(f, " had/would"); }
ain't          { fprintf(f, " am/is/are/has/have not"); }
Ain't          { fprintf(f, " Am/Is/Are/Has/Have not"); }
```

Figura 6.2: Contrações com múltiplas opções.

Em seguida procurou-se a cobertura de contrações não tão frequentes, como *gonna*, *wanna* e outras sugeridas por uma breve pesquisa na web, e passou-se à segunda etapa da resolução do problema: a listagem dos verbos utilizados no infinitivo no texto ao qual se aplica o processador FLEX. A definição da estrutura onde seriam armazenados os verbos, ao longo do processamento da análise do ficheiro de *input*, foi o primeiro passo. A escolha recaiu sobre as árvores da biblioteca GLib, à semelhança do que acontecera na resolução da solução para o problema dos processadores de Named Entities, porque pretendia-se evitar que um mesmo verbo fosse introduzido na estrutura mais do que uma vez. Assim, cada chave de um nodo na árvore representa um verbo e o valor o número de ocorrências desse mesmo verbo, ou seja, sempre que se encontra um verbo repetido, este não é inserido, apenas se incrementa o valor do nodo. Na Fig. 6.4 está a função responsável pela inserção na árvore. Nota também para a contagem do número de verbos identificados, através de uma variável *n*, para controlo do número de verbos repetidos encontrados e, assim, não introduzidos - a diferença entre *n* e o número de elementos na árvore resulta no número de verbos repetidos identificados pelo processador de texto.

```
GTree *verbs;
int n = 0;
```

Figura 6.3: Estrutura usada e variável auxiliar.

```
void insert(GTree *tree, char *key){
    int *value = g_tree_lookup(tree, key);
    if (value != NULL){
        (*value)++;
    }
    else{
        value = malloc(sizeof(int));
        *value = 1;
    }
    g_tree_insert(tree, key, value);
}
```

Figura 6.4: Função de inserção na árvore de verbos.

Chegou, então, a altura de pensar em que situações se usam verbos no infinitivo. Tal como consta no enunciado, os casos mais óbvios serão aqueles em que o verbo é usado com *to* atrás ou quando surgem associados a palavras como *do* ou *did*, porém existem outras situações que o grupo procurou cobrir - a seguir a contrações como *gonna* e *wanna* ou a seguir a frases interrogativas com verbos modais (*should*, *may*, *must*, etc.), por exemplo, a frase requer um verbo no infinitivo.

QUESTIONS	[Dd]o [Dd]oes [Dd]id [Ss]hall [Ss]hould [Mm]ay [Mm]ight [Cc]an [Cc]ould [Ww]ill [Ww]ould [Oo]ught [Mm]ust
NEGATIVE	[Dd]on't [Dd]oesn't [Dd]idn't [Ss]houldn't [Mm]ightn't [Cc]ouldn't [Mm]ustn't [Ww]ouldn't
SUBJECT	[Ii] [Yy]ou [Hh]e [Ss]he [Ww]e [Tt]hey [A-Z]{1}[a-z]+
ALPHA	[a-z]+

Figura 6.5: Cobertura de verbos nas questões.

Nesta questão dos verbos há novamente um senão. Começando pelo facto de a palavra *to* não ser usada apenas como antecessor de verbos, mas também como preposição, cria logo a necessidade de excluir diversos casos. Uma frase exemplo: *I want to give a present to Anthony's brother.* - nesta frase temos dois *to*, um que precede o verbo *want* e outro que precede um complemento indireto. Apesar de ser difícil cobrir todas as situações, através da representação das exceções (Fig. 6.5) e das situações em que se segue letra maiúscula, números ou sinais de pontuação a um *to*, consegue-se que a maioria dos *to* seja corretamente identificada como verbo ou preposição.

Por fim, para aumentar a área de atuação do processador FLEX, decidiu-se colocar algumas abreviaturas, para além de contrações, no lote de expressões regulares filtradas.

```

{QUESTIONS}\ {SUBJECT}\ {ALPHA}      { fprintf(f, "%s", yytext);
                                        strtok(yytext, " ");
                                        strtok(NULL, " ");
                                        char* v = strdup(strtok(NULL, " "));
                                        insert(verbs, v);
                                        n++; }

[Cc]an't\ {SUBJECT}\ {ALPHA}          { strtok(yytext, " ");
                                        char* subject = strtok(NULL, " ");
                                        char* v = strdup(strtok(NULL, " "));
                                        fprintf(f, "%cannot %s %s", yytext[0], subject, v);
                                        insert(verbs, v);
                                        n++; }

[Ww]on't\ {SUBJECT}\ {ALPHA}          { strtok(yytext, " ");
                                        char* subject = strtok(NULL, " ");
                                        char* v = strtok(NULL, " ");
                                        fprintf(f, "%cill not %s %s", yytext[0], subject, v);
                                        insert(verbs, v);
                                        n++; }

{NEGATIVE}\ {SUBJECT}\ {ALPHA}        { char* rep = strdup(yytext);
                                        char* neg = strtok(yytext, "n't");
                                        strtok(rep, " ");
                                        char* subject = strtok(NULL, " ");
                                        char* v = strtok(NULL, " ");
                                        fprintf(f, "%s not %s %s", neg, subject, v);
                                        insert(verbs, v);
                                        n++;
                                        free(rep); }

```

Figura 6.6: Inclusão dos verbos, usando as definições acima.

```

SIGNAL      [.,;:-?!\n\t]
EXCEPTIONS  me|you|him|it|us|them|my|your|his|her|its|our|their|mine|yours|hers|ours|theirs|the|a|an|all|many|this|that
NUMBERS     one|two|three|four|five|six|seven|eight|nine|ten|eleven|twelve|thirteen|fourteen|fifteen|sixteen|seventeen|
            eighteen|nineteen|twenty|thirty|forty|fifty|sixty|seventy|eighty|ninety

```

Figura 6.7: Algumas palavras que, não sendo verbos, podem suceder a um *to*.

```

[Tt]o\ {EXCEPTIONS}      { fprintf(f, "%s", yytext); }
[Tt]o\ {A-Z}             { fprintf(f, "%s", yytext); }
[Tt]o\ {0-9}+            { fprintf(f, "%s", yytext); }
[Tt]o\ {NUMBERS}         { fprintf(f, "%s", yytext); }
[Tt]o\ {SIGNAL}          { fprintf(f, "%s", yytext); }
[Tt]o\ {ALPHA}           { fprintf(f, "%s", yytext); }

```

Figura 6.8: Casos em que o *to* não antecede um verbo.

```

[Bb][Tt][Ww]          { fprintf(f, "%cy %che %cay", yytext[0], yytext[1], yytext[2]); }
[Bb][Ff][Ff]          { fprintf(f, "%cest %criend %corever", yytext[0], yytext[1], yytext[2]); }
[Ii][Ll][Yy]          { fprintf(f, "%c %cove %cou", yytext[0], yytext[1], yytext[2]); }
[Ii][Dd][Gg][Aa][Ff] { fprintf(f, "%c %con't %cive %cuck", yytext[0], yytext[1], yytext[2], yytext[3], yytext[4]); }
[Jj][Kk]              { fprintf(f, "%cust %cidding", yytext[0], yytext[1]); }
[Ll][Mm][Aa][Oo]      { fprintf(f, "%caughing %cy %css %cff", yytext[0], yytext[1], yytext[2], yytext[3]); }
[Ll][Oo][Ll]          { fprintf(f, "%caughing %cut %coud", yytext[0], yytext[1], yytext[2]); }
[Nn][Pp]              { fprintf(f, "%co %crobblem", yytext[0], yytext[1]); }
[Oo][Mm][Gg]          { fprintf(f, "%ch %cy %cod", yytext[0], yytext[1], yytext[2]); }
[Ww][Tt][Ff]          { fprintf(f, "%chat %che %cuck", yytext[0], yytext[1], yytext[2]); }

```

Figura 6.9: Abreviaturas incluídas.

Capítulo 7

Apresentação das Queries e Resoluções - TEMA 2

7.1 Queries de Resolução Obrigatória

Naturalmente, como queries obrigatórias procurou-se responder à listagem dos verbos usados no infinitivo e à remoção de contrações, típicas do inglês corrente, tal como aponta o enunciado.

7.2 Queries Adicionais

- a) Adicional 1 Adição de abreviaturas, para respetiva expansão, à filtragem do processador de texto;
- b) Adicional 2 Contagem do número de repetições de cada verbo usado no infinitivo.

Capítulo 8

Apresentação de Resultados

Neste capítulo serão apresentados exemplos de utilização.

8.1 Execução

A conversão de inglês corrente envolve as seguintes etapas:

1. Colar o texto a converter na caixa de texto;
2. Confirmar o armazenamento num ficheiro;
3. Correr o programa FLex;
4. Ver resultados.

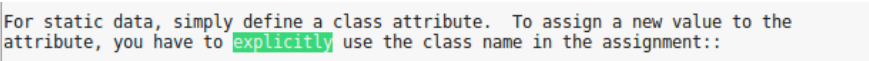
8.2 Exemplos

Agora apresentar-se-ão os resultados da aplicação quando recebeu como *input* os seguintes textos:

1. FAQ-Python-En.txt, fornecido pela equipa docente aquando do lançamento deste enunciado;
2. OUTRO

Relativamente ao primeiro caso, a quantidade de contrações é reduzida apesar do tamanho do texto. O processador alcança bons resultados, quer nos verbos listados quer no texto sem contrações, no entanto sobressaem dois erros recorrentes:

1. a colocação por extenso da contração *'s* quando esta atua enquanto símbolo de posse (*possessive 's*). Este é o tipo de erro difícil de combater, visto que *'s* pode ser usado como contração de *is* ou *has*, mas também no contexto de posse (*Anthony's house is red.*);
2. a inclusão de advérbios na listagem de verbos, resultante de frases do género de *He has to seriously consider it!*, onde o advérbio toma a posição do verbo, isto é, a seguir ao *to*, perturbando a eficácia do processador.



```
For static data, simply define a class attribute. To assign a new value to the attribute, you have to explicitly use the class name in the assignment::
```

Figura 8.1: Advérbio na listagem de verbos.

• elements x1
• emulate x1
• examine x1
• explicitly x1
• extend x1
• faster x1

Figura 8.2: Excerto da listagem de verbos.

Capítulo 9

Conclusão

Terminado o projeto, salientamos a utilidade e facilidade do uso do Flex na criação de filtros a aplicar a textos, algo que se tornaria bem mais complexo se contruíssemos e utilizássemos um programa em C ou noutra linguagem de programação mais comum.

Mais uma vez, como no TP1a - GAWK, é de destacar a criação de uma página HTML, que, apesar de já ter sido também criada para o trabalho referido, permanece algo novo entre o grupo e ainda assim com resultados, na nossa perspectiva, bastante satisfatórios.

Quanto a trabalho futuro, seria lógico, no caso do Exercício 2.1., continuar a desenvolver casos para tentar aproximar da perfeição o filtro. Da mesma forma, falando agora sobre o trabalho futuro para o Exercício 2.4., será lógico que o próximo passo seja a criação de mais Tags que possam ser incluídas no ficheiro XML e identificadas pelo filtro Flex.