Normalizador de Autores em BibTex
&
Processador de Inglês corrente em Flex

Processamento de Linguagens
(3º ano de Curso)
Trabalho Prático nº1 - Parte B
Relatório de Desenvolvimento

José Silva	Pedro Cunha	Gonçalo Moreira
(A74601)	(A73958)	(A73591)

23 de Abril de 2017

Resumo

Documentação do segundo trabalho prático da unidade curricular de "Processamento de Linguagens". O principal foco incide em utilizar o analisador léxico FLEX, para desenvolver processadores de texto. Simplifica o trabalho que seria necessário utilizando diretamente a linguagem de programação C, facilitando o processo. Demonstrando e documentando as soluções propostas pelo grupo de trabalho para os dois problemas escolhidos, termina-se o relatório com uma análise argumentativa sobre a eficiência dessas mesmas soluções.

Conteúdo

Capítulo 1

Introdução

Tal e qual como foi descrito no anterior trabalho prático, os últimos anos trouxeram consigo a inevitabilidade de processar cada vez mais texto. Extrair e editar determinadas linhas, onde certos padrões são bastante evidentes, tornou-se assim algo fundamental. Justificam-se assim as várias ferramentas criadas para o efeito, dentro das quais se destaca o FLEX. Assumindo-se como um programa capaz de gerar analisadores lexicais, juntamente com o uso de expressões regulares e as funcionalidades disponibilizadas pela linguagem de programação C, a utilização do FLEX permite a resolução dos vários problemas relacionados com o que anteriormente foi descrito.

Neste segundo trabalho prático da unidade curricular de "Processamento de Linguagens", através dos meios descritos anteriormente, vão ser realizados dois filtros de texto capazes de processar algumas das características da língua inglesa e de normalizar certos componentes de um ficheiro no formato BibTex. Para além disso, são realizados alguns extras como, por exemplo, a criação de um grafo representativo das ligações entre diversos autores e um página HTML onde são apresentados os diversos verbos no infinitivo não flexionado resultantes do processamento de um texto em inglês.

Estrutura do Relatório

No capítulo 1 faz-se uma pequena introdução ao problema e às ferramentas utilizadas para a resolução deste. Para além disso, é descrita de uma forma breve a estrutura do relatório.

No capítulo 2 faz-se uma análise breve mas mais detalhada do problema escolhido pelo grupo de trabalho.

No capítulo 3 é descrito de uma forma sumarizada o procedimento utilizado para solucionar as várias questões propostas pelos enunciados.

No capítulo 4 são apresentados alguns testes e respetivos resultados para comprovar o respectivo funcionamento das soluções apresentadas.

Finalmente, no capítulo 5 termina-se o relatório com uma síntese do que foi dito, as conclusões e o trabalho futuro.

Capítulo 2

Análise e Especificação

2.1 Normalizador de Autores em BibTex

2.1.1 Descrição informal do problema

É fornecido um ficheiro em BibTex(como input), com várias entradas e diversos autores e editores por entrada. Pretende-se que se desenvolva um "Normalizador"para ler esse mesmo ficheiro e gerar um ficheiro equivalente (em BibTex)com os nomes de todos os editores e autores normalizados. Também se pretende converter todos os carateres com acentos explícitos em caracteres portugueses. A forma de normalização dos nomes e dos acentos explícitos é apresentada em detalhe a baixo, na Especificação dos Requisitos.

2.1.2 Especificação dos Requisitos

Dados

Como já foi referido, é fornecido um ficheiro em BibTex com várias entradas e diversos autores e editores por entrada. Este ficheiro contém, em cada entrada, diversos fields que poderão ou não incluir os fields author e editor. Podem existir diversos tipos de entrada, com número e tipos diferentes de fields.

Cada field pode ocupar uma ou mais linhas, dependendo da informação que o mesmo representa. Na maioria dos casos de editor e author ocupa apenas uma, mas não é regra. Quando existem muitos autores e/ou editores é normal que sejam necessárias mais linhas. Podem também existir acentos explícitos em qualquer field que tenha texto, incluindo nos nomes de editores e autores.

Pedidos

Como primeiro requisito, é pedido que todos os acentos explícitos(e cedilhas) sejam convertidos para caracteres portugueses, exemplos:

Anast\'acia ou Anast{\'a}cia deve ser convertido em: Anastácia.
Gon\c{c}alo ou Gon{\c{c}}alo deve ser convertido em: Gonçalo.
Também é pedido que os nomes de autores e editores sejam normalizados, todos têm que apresentar a mesma forma. A forma requisitada é a seguinte: "Apelido, N1. N2."em que N1 e N2 são possíveis nomes próprios e Apelido é, obviamente, o apelido(ou apelidos em alguns casos). Salienta-se que também é necessário que sejam todos da forma "author/editor = { ... }", por isso casos que usem aspas ou o número errado de espaços também devem ser corrigidos.
Exemplo de normalização:
author="Martini, Ricardo G. and Ara\'ujo, Cristiana and Almeida, Jos\'e Jo\ ao and Henriques, Pedro"
Deve ficar:
author = {Martini, R. G. and Araújo, C. and Almeida, J. J. and Henriques, P.}

Por fim, é solicitado que se use a linguagem Dotty e o processador dot para criar um grafo com todos os autores que colaboram com um dado autor.

2.2 Processador de Inglês corrente

O problema proposto pela equipa docente divide-se em duas partes. É fornecido um ficheiro exemplo contendo texto, em inglês, com vários exemplos das multiplas contrações caracteristicas desta lingua e com vários verbos numa determinada f orma nominal.

Numa primeira parte, pretende-se que se desenvolva um "Normalizador"capaz de gerar um ficheiro equivalente, mas que modifique as várias contrações existentes na gramática inglesa na sua forma normal. Numa segunda parte pretende-se gerar um ficheiro, em formato HTML, que possua todas as ocorrências de verbos no infinitivo não lexionado presentes no ficheiro de input.

2.2.1 Especificação dos Requisitos

Dados

O ficheiro fornecido para a execução das duas etapas, descritas anteriormente, apresenta um formato comum. O programa deve estar preparado para lidar com qualquer tipo de ficheiro, seja ele no formato BibTex, html, ou qualquer outro. As várias contrações podem estar definidas tanto na forma negativa como positiva e encontrar-se em qualquer zona do ficheiro. À semelhança, os verbos e os padrões que os acompanham estão sujeitos às mesmas características.

Pedidos

Como primeiro requisito, é pedido que as contrações sejam convertidos para a sua forma normal. Os casos não são uniformes como podemos ver pelos seguintes exemplos:

I'm - I am

W're - We are

Como segundo requisito, é pedido que sejam retirados, para um ficheiro diferente, todos os verbos no infinitivo não lexionado. Aqui, são três os padrões conhecidos:

- "To accumulate.". A forma verbal é precedida pela forma "to".

- "I might go". A forma verbal é antecedido por can, could, shall, should, will, would, may, might, bem como as suas formas negativas.

- "Do you want to go?". Na forma interrogativa o verbo é precedido por 'do,does, did,can,could,shall, should, will, would, may, might + qualquer outra palavra'. Aqui também, tal como nos anteriores exemplos, as formas negativas.

Capítulo 3

Concepção/desenho da Resolução

3.1 Normalizador de Autores em BibTex

3.1.1 Estruturas de Dados

Depois do levantamento de requisitos, chegou-se à conclusão que para o primeiro não seria necessário guardar em memória nenhum tipo de dados, bastando substituir os caracteres encontrados.

Já no segundo requisito, dada a ordem que os nomes aparecem, é necessário guardar os nomes próprios (num array de char*) de forma a enviar para o ficheiro de output apenas quando for encontrado o apelido.

Quanto à criação do grafo, é necessário, mais uma vez, um array de char*. Desta vez o objetivo é guardar os nomes que já apareceram na mesma tag de author/editor de forma a associar todos aos nomes novos que estão na mesma linha e são descobertos.

3.1.2 Algoritmos

Seguindo a abordagem anterior, os algoritmos serão descritos requisito por requisito. No caso dos acentos explícitos, não existe nenhum algoritmo complexo para resolver o problema, pois neste caso é apenas necessário substituir sequências de caracteres que aparecem por um caracter português correspondente. (Ficheiro convAcentos, apêndice A)

Para efetuar a normalização dos nomes é mais complexo, pois os nomes podem aparecer em várias formas e é necessário que sejam todos apresentados da mesma forma. A primeira fase passa por encontrar todos os fields com "author="e "editor=", caso sejam utilizadas aspas é necessário trocar por chavetas. Todos os restantes fields são imprimidos como estão. É também essencial(apenas quando estamos perante um dos fields pretendidos) encontrar todos os nomes próprios e guardar cada um deles na estrutura de dados já referida. O objetivo desta última abordagem é encontrar o apelido logo depois e imprimir com todos os nomes próprios encontrados.

Embora a maioria dos casos funcionem com o que foi descrito, existem casos em que a situação é diferente. Esses casos aparecem numa forma intermédia entre estes descritos e a forma pretendida. Apresentam-se com os apelidos já divididos dos nomes próprios, mas ainda com os nomes próprios na forma completa. Nestes casos, são selecionados todos os nomes próprios e apelidos na mesma expressão regular para depois ser tratada a informação obtendo a forma final que é solicitada.

Finalmente, temos o caso da criação do grafo. Para criar o grafo será utilizado o ficheiro de output dos requisitos anteriores de forma a facilitar o tratamento dos nomes(justificação em mais detalhe no próximo capítulo). É importante criar uma função de divisão de strings por palavra, pois neste caso é necessário dividir os nomes por "and ". O uso desta função será similar ao da função strtok da biblioteca string.h. Com esta função torna-se fácil o processo, bastando encontrar todos os nomes separados por "and "ou então sozinhos quando há só um. Depois basta imprimir os nomes na forma necessária seguindo a sintaxe Dot para um ficheiro.

3.2 Processador de Inglês corrente

3.2.1 Estruturas de Dados

Pensando nos requisitos para este projeto é fácil de perceber que as duas partes do trabalho prático vão ter abordagens distintas, no que diz respeito às estruturas de dados. Na primeira parte, onde queremos como output um ficheiro semelhante àquele que recebemos como input, não precisamos de qualquer estrutura de dados auxiliar já que as funcionalidades do flex, da linguagem c e das expressões regulares nos permitem fazer face ao problema.

Por outro lado, no que diz respeito à passagem dos verbos no infinitivo não lexionado para um ficheiro output sem qualquer semelhança ao de input, é necessária uma estrutura de dados. Tal e qual como nas aulas práticas da unidade curricular, fizemos uso da biblioteca GLIB. Para o efeito, escolhemos a implementação de listas ligadas disponibilizada por esta biblioteca. Os factores que mais pesaram na escolha da estrutura de dados foram o facto de pudermos inserir uma quantidade indeterminada de itens e a disponibilização de uma função de inserção ordenada.

Ao longo da resolução do problema por vezes foi necessário apenas retirar uma certa parte de texto, do total daquilo que é adquirido através do uso de uma certa expressão regular. Assim, foi necessário utilizar a função strtok. Por consequência, definimos um array e uma string capazes de guardar informação temporária e de auxiliar a função descrita anteriormente. Finalmente, foi utilizada uma string para que a impressão de verbos repetidos não se realizasse.

3.2.2 Algoritmos

No que diz respeito à primeira parte do exercício prático, respeitante às contrações gramaticais, três abordagens distintas tiveram que ser realizadas. Numa primeira abordagem, respeitante por exemplo à contração "I'm", a expressão regular utilizada faz matching apenas com o "' "e tudo o que lhe sucede, já que são os únicos elementos que devem ser alterados por a expressão correspondente. Tudo o que antecede, permanece na mesma. O mesmo acontece para algumas das formas negativas como, por exemplo, "Don't". No que diz respeito a contrações como "Won't"ou "Can't", não existe um padrão especifico e por isso a expressão regular que faz matching deve ser alterada na sua totalidade.

Por outro lado, no que diz respeito à segunda parte do exercício prático, o nível de complexidade aumenta. Quando o verbo no infinitivo é precedido por "to", a expressão regular deve fazer matching com o "to", seguido de um espaço, e finalmente seguido do verbo que desejamos. Assim, na altura de guardar o verbo na estrutura de dados devemos redirecionar o apontador três caracteres à frente de modo a que a palavra "to"e o consequente espaço não sejam guardados. Quando o verbo no infinitivo é precedido por "can/could...", a expressão regular é semelhante à descrita anteriormente. Mas se no caso anterior era sempre conhecido o tamanho da palavra que precede o verbo, neste caso em concreto, não. Assim, é necessário utilizar a função strtok de modo a dividir as palavras que fizeram matching por espaços. Feito isto, os passos são semelhantes aos anteriores, com o verbo a ser inserido na estrutura de dados. Finalmente, quando estamos perante uma forma interrogativa, ao contrário dos dois casos anteriores, o verbo vai encontrar-se com maior probabilidade na segunda palavra que sucede às expressões "do/did/does". O procedimento é então bastante semelhante ao caso anterior.

Como sabemos, nem tudo retirado pelas expressões regulares corresponde a verbos. Assim, foram definidas uma série de excepções correspondentes a pronomes, advérbios, alguns padrões não frequentes em verbos no infinitivo, etc de forma a tentar evitar que o que seja imprimido no ficheiro output corresponda ao desejado.

Na parte correspondente à main é tratado tudo o que está relacionado com a estrutura de um ficheiro HTML. Para além disto, são impressos para o ficheiro de output os vários verbos presentes na estrutura de dados, excluindo os repetidos.

Capítulo 4

Codificação e Testes

4.1 Normalizador de Autores em BibTex

4.1.1 Alternativas, Decisões e Problemas de Implementação

De forma a facilitar o procedimento, foi decidido que existiriam 3 programas para solucionar o problema. Assim, existe o programa que resolve os acentos explícitos (convAcentos.fl), o programa que normaliza os nomes dado o ficheiro de output de convAcentos(normNomes.fl) e ainda o programa que gera o grafo dado o ficheiro de output de normNomes. Uma alternativa seria fazer tudo no mesmo programa, mas isso dificultaria o processo porque os nomes não estariam todos na mesma forma para o grafo, e os matches de acentos entrariam em conflito com os matches de nomes. Para toda esta execução existe um Makefile para facilitar. Quanto ao grafo gerado foi utilizado um grafo não orientado de forma a respresentar as relações entre autores que colaboraram.

Um dos problemas de implementação foi a necessidade de implementar a função multi_tok (no ficheiro genGraph, linhas 62 a 82) com o objetivo de separar string por uma dada palavra, utilização similar a strtok. Outro problema durante a implementação foi o facto dos nomes aparecerem em diversas formas, este foi resolvido seguindo o procedimento descrito no capítulo anterior, mas houve ainda o cuidado com alguns nomes que têm "de, dos, da", que era ignorado. Foi também necessário ter um cuidado adicional com nomes que começam com letra acentuada, pois neste caso ocupa mais do que um char normal. A resolução destes casos foi feita através de verificação do código ASCII dessas mesmas letras.

4.1.2 Testes realizados e Resultados

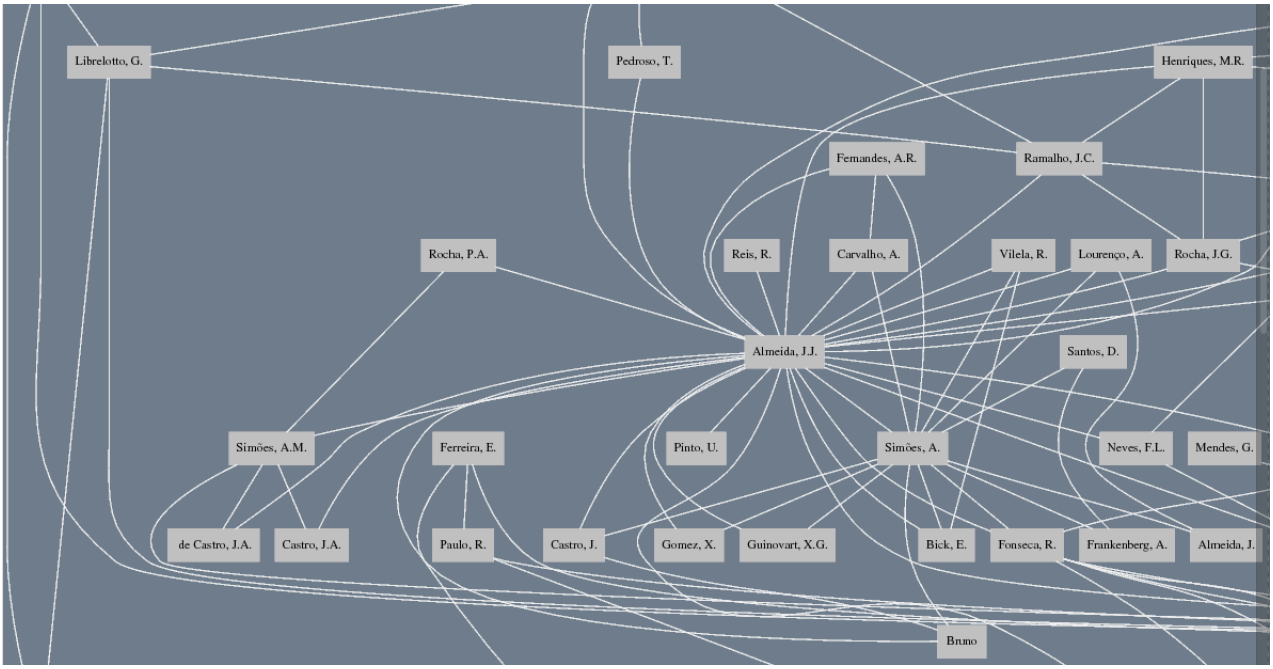
4.1.3 Execução para o ficheiro dado no enunciado(exemplo-utf8.bib)

Excertos do ficheiro gerado

```
@techreport{jj95,
    author = {Almeida, J.J.},
    title = "{NLlex} -- a tool to generate lexical analysers for natural language",
    institution = umdi,
    year   = 1995,
    number = "UM-DI-95.04",
    url = "http://natura.di.uminho.pt/~jj/pln/nllex.ps.gz",
  keyword ="jspell,morphology,lex,PLN,nllex",
  }

  @techreport{Barbosa95,
   author = {Barbosa, L.S. and Almeida, J.J.},
   title  = "System Prototyping in \textsc{Camila}",
   year = 1995,
   number = "DI-CAM-95:11:1",
   institution="University of Minho",
   note = "Lecture notes for the System Design Course,
           Computer System Engineering, University of Bristol",
   url="http://www.di.uminho.pt/~lsb/pub_camila/LNcam.ps.gz",
   keyword ="Camila, formal specification",
  }

  @techreport{Barbosa95a,
   author = {Barbosa, L.S. and Almeida, J.J.},
   title  = "\textsc{Camila}: A reference Manual",
   year = 1995,
   number = "DI-CAM-95:11:2",
   institution="University of Minho",
   url="http://www.di.uminho.pt/~lsb/pub_camila/RMcam.ps.gz",
   keyword ="Camila",
  }

  @techreport{BA97a,
    keyword = "Formal Methods, Prototyping, Camila",
    author = {Barbosa, L.S. and Almeida, J.J.},
    title = "Systems Prototyping in \textsc{Camila}",
    type = "{Lecture Notes for the Bristol Course  (1st ed.  1995)}",
    year = 1998,
    institution = "DI (U. Minho)",
    number = "DI-CAM-95:11:1:v98"
    }
```

```
@inproceedings{LRHGT08,
  author = {Librelotto, G.R.
       and Ramalho, J.C.
       and Henriques, P.R.
       and Gassen, J.B.
       and Turchetti, R.C.},
  title={A Framework to specify, extract and manage Topic Maps driven by ontologie},
  booktitle={SIGDPC'08 -- 26th ACM International Conference on Design of Communication},
  address={Lisboa, Portugal},
  year={2008},
  month={Sep},
}

@inproceedings{LGFSSAH08,
  author = {Librelotto, G.R.
       and Gassen, J.B.
       and Freitas, L.O.
       and Silveira, M.C.
       and Silva, F.L.
       and Augustin, I.
       and Henriques, P.R.},
  title={Uma Ontologia aplicada a um Ambiente Pervasivo Hospitalar},
  booktitle={CAPSI'08 -- 8º Conferência da Associação Portuguesa de Sistemas de Informação},
  address={Setubal, Portugal},
  year={2008},
  month={Oct},
}

@InProceedings{LMMVRH08,
  author = {Librelotto, G.R.
       and Martins, M.
       and Machado, H.
       and Vizzotto, J.
       and Ramalho, J.C.
       and Henriques, P.R.},
  title =     {Generating a Semantic Network for PubMed},
  booktitle = {XATA08 --- {XML} : Aplicações e Tecnologias Associadas, Évora, Portugal},
  year =     {2008},
  month =     {February}
}
```

Parte do grafo gerado

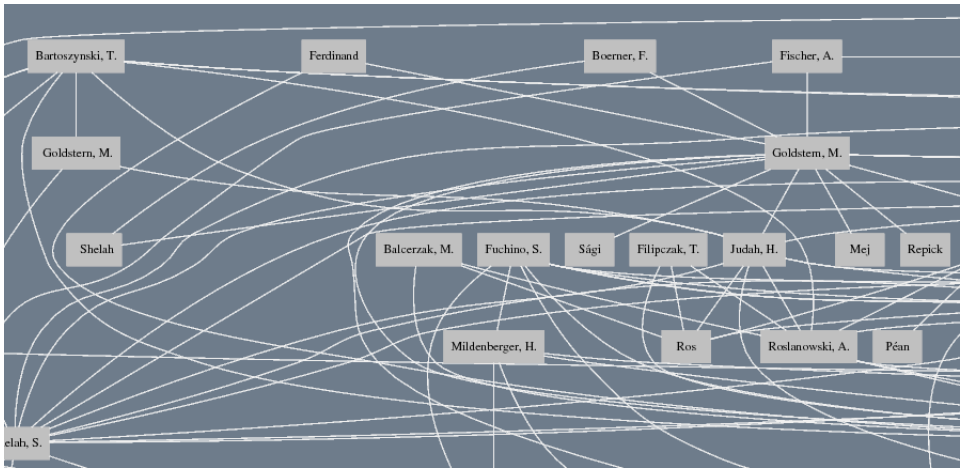


4.1.4 Execução para um ficheiro obtido pelo grupo de trabalho(listb.bib)

Excertos do ficheiro gerado

```
@article{MoSh:27,
author = {Moran, G. and Shelah, S.},
ams-subject = {(90D05)},
journal = {Israel Journal of Mathematics},
review = {MR 47:10084},
pages = {442--449},
title = {{Size direction games over the real line. III}},
volume = {14},
year = {1973},
},

@article{Sh:28,
author = {Shelah, S.},
ams-subject = {(02B15)},
journal = {Israel Journal of Mathematics},
review = {MR 49:20},
pages = {282--300},
title = {{There are just four second-order quantifiers}},
volume = {15},
year = {1973},
},

@article{Sh:29,
author = {Shelah, S.},
ams-subject = {(05A05)},
journal = {Journal of Combinatorial Theory. Ser. A},
review = {MR 48:10824},
pages = {199--208},
title = {{A substitute for Hall's theorem for families with infinite
 sets}},
volume = {16},
year = {1974},
},

@incollection{MzSh:30,
author = {McKenzie, R. and Shelah, S.},
booktitle = {Proceedings of the Tarski Symposium (Univ. California,
 Berkeley, Calif., 1971)},
ams-subject = {(02H15)},
review = {MR 50:12711},
pages = {53--74},
publisher = {Amer. Math. Soc., Providence, R.I},
series = {Proc. Sympos. Pure Math.},
```

Parte do grafo gerado



4.2 Processador de Inglês corrente

4.2.1 Alternativas, Decisões e Problemas de Implementação

Tal como sugerido pela segunda parte do primeiro exercício prático, o ficheiro output deveria de estar no formato HTML. Tal e qual como no primeiro trabalho prático decidimos que o ficheiro deveria utilizar também estilos escritos em css e incluir um script(em javascript) para tornar possível o aparecimento de listas escondidas. Também foi utilizada a font-awesome para juntar icons que refletem o conteúdo de cada linha apresentada. O objetivo do uso destes recursos é tornar o conteúdo do ficheiro apelativo. Para além disso, poderíamos ter optado por outro tipo de estruturas de dados, como por exemplo, árvores binárias.

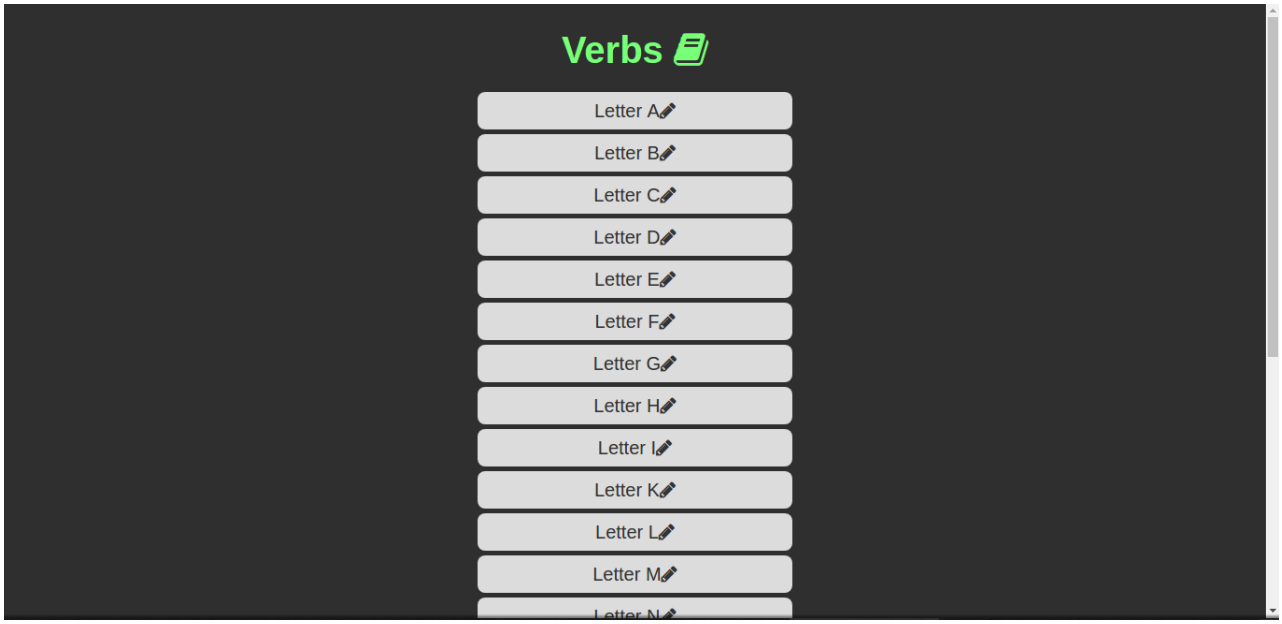
No que diz respeito à implementação, achamos que a o modo de resolução do problema nunca poderia ser muito diferente daquele realizado. Seria possível acrescentar mais casos que restringissem o aparecimento de outras palavras que não fossem verbos. Mas, por outro lado, a adição de mais exceções poderia levar igualmente à supressão de verbos que estariam correctos a ser apresentados no ficheiro de output.

4.2.2 Testes realizados e Resultados

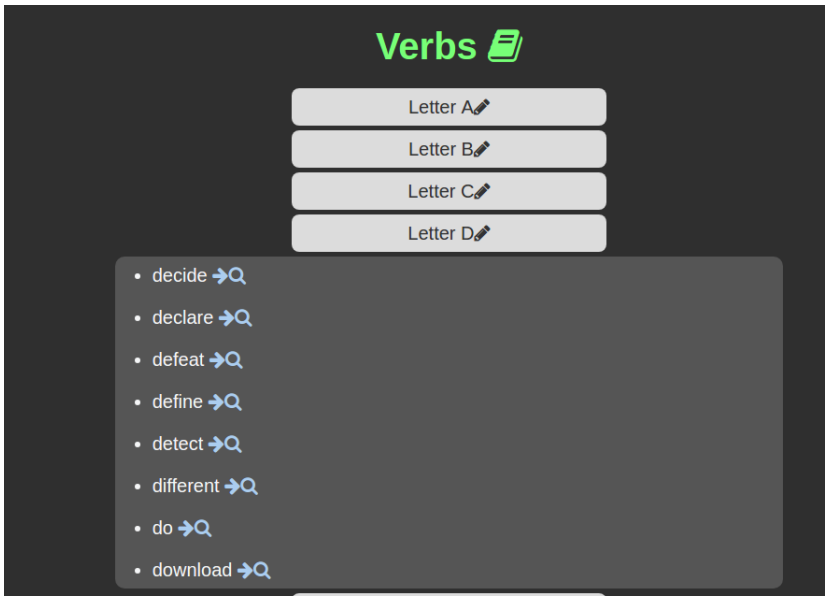
4.2.3 Execução para ficheiro dado no enunciado(FAQ-Phyton-EN.txt)

Html obtido

Lista antes de escolher letra

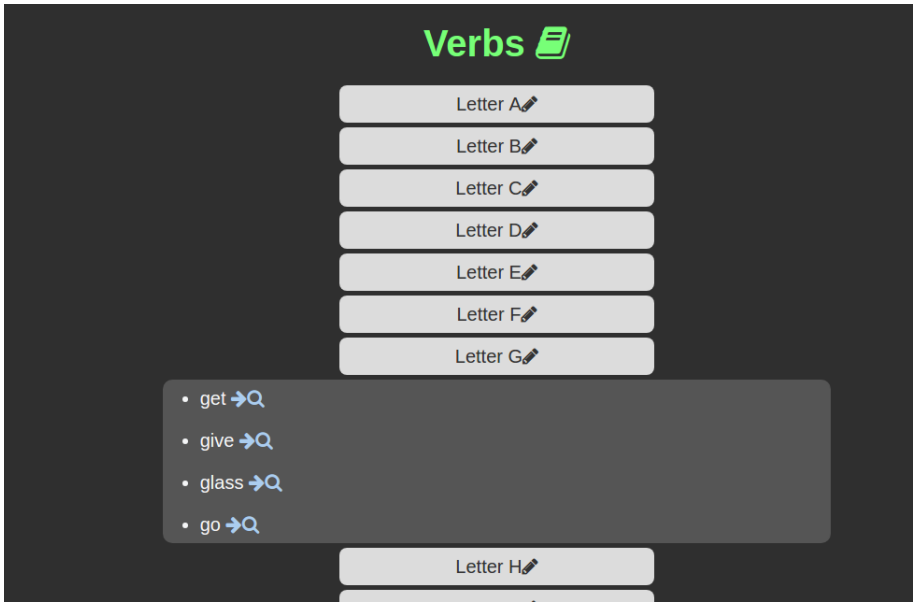


Lista depois de escolher letra

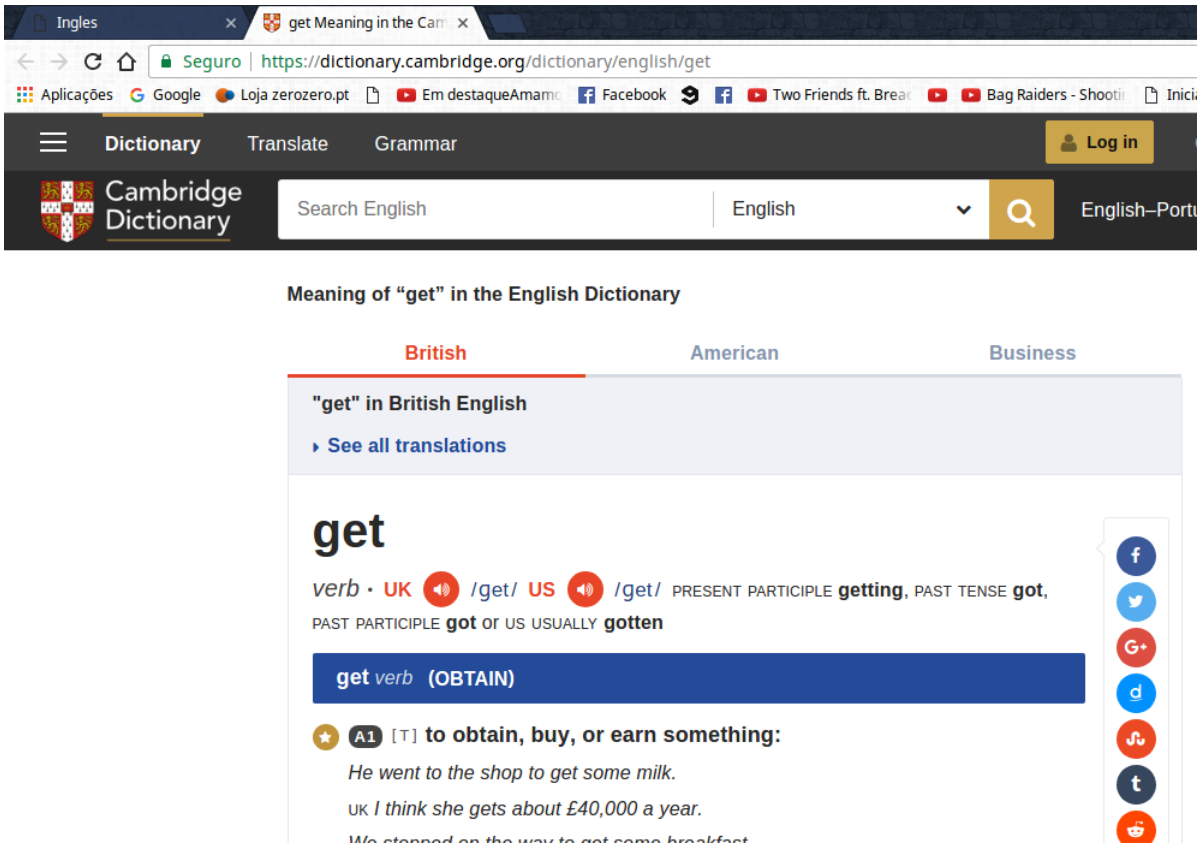


4.2.4 Execução para um ficheiro obtido pelo grupo(potter.txt)

Html obtido



Clicando na lupa de pesquisa para get



Capítulo 5

Conclusão

Neste projeto foi possível atingir os objetivos propostos, relativos à criação de um Normalizador de Autores em BibTex, utilizando Expressões Regulares para a definição de padrões, desenvolvendo Processadores de Linguagens Regulares para a filtragem e transformação de textos e recorrendo ao Analisador Léxico FLex para gerar filtros de texto em C.

Com a criação de tal Normalizador foi possível a partir de publicações, normalizar e reescrever acentos em caracteres portuguese assim como os campos relativos aos editores e editores de modo a serem reescritos de acordo com uma estrutura especifica. Após a normalização das publicações seria gerado um grafo com todos os autores e ligações de colaboração entre eles, utilizando a linguagem Dotty e processador dot para tal criação assim como o uso de expressões regulares e analisadores léxicos para a criação do ficheiro de input do processador dot.

Para além da criação deste Normalizador de Autores também foi possível a realização de um Processador de Inglês corrente, no qual seria possível a expansão de contrações da língua inglesa a partir de um dado texto e gerar um ficheiro html com uma lista ordenada de todos os verbos ingleses no infinitivo presentes no texto.

Com estes dois projetos foi possível estabelecer padrões de frases nos textos utilizados através de Expressões Regulares assim como identificar as ações semânticas relativas aos padrões de frases estabelecidos. Com o estabelecimento destes padrões foi possível a criação de um Filtro de Texto capaz de os reconhecer e , recorrendo ao Gerador Flex, transformar os textos de acordo com os requisitos.

Apêndice A

Código do Programa para Normalizador de Autores em BibTex

Lista-se a seguir o código do programa que foi desenvolvido.

../convAcentos.fl

```
%option noyywrap


%%
    FILE *output  = fopen("outAc.bib", "w");

    /* Acentuação */
\{?\\'A\}?     { fprintf(output, "Á"); }
\{?\\'E\}?     { fprintf(output, "É"); }
\{?\\'I\}?     { fprintf(output, "Í"); }
\{?\\'O\}?     { fprintf(output, "Ó"); }
\{?\\'U\}?     { fprintf(output, "Ú"); }
\{?\\~A\}?     { fprintf(output, "Ã"); }
\{?\\'a\}?     { fprintf(output, "á"); }
\{?\\'e\}?     { fprintf(output, "é"); }
\{?\\'i\}?     { fprintf(output, "í"); }
\{?\\'o\}?     { fprintf(output, "ó"); }
\{?\\'u\}?     { fprintf(output, "ú"); }
\{?\\~a\}?     { fprintf(output, "ã"); }
\{?\\c\{c\}\}? { fprintf(output, "ç"); }
\{?\\C\{C\}\}? { fprintf(output, "Ç"); }

    /*
        Print o que sobrou para não se
        perder info do ficheiro original
    */
.|\n { fprintf(output, "%s", yytext); }

%%


int main (int argc, char* argv[]) {
    FILE *input;
    input = fopen(argv[1], "r");
    yyin = input;
    yylex();
    return 0;
```

```
}
```

../normNomes.fl

```
%option noyywrap
%s LIMPAR

%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
%}

L       [a-z]|ú|ù|á|à|ç|é|ê|è|õ|ó|ò|ã|ì|í|\.
LM      [A-Z]|Ú|Ù|Á|À|Ç|É|É|Õ|Ó|Ò|Ã|Ì|Í
T       (author|editor)
AUX     (([\ ]+and)|[\}\"\ ]*))
N       (da|dos|de)[\ ]

%%
    int i = 0, j = 0, x;
    int aspas = 0;
    int otherTag = 0;
    char **nomesProprios, *token;
    char apelido[64], *nomes;

    nomesProprios = malloc(sizeof(char *) * 20);
    FILE *output  = fopen("out.bib", "w");

    for(x = 0; x < 20; x++)
        nomesProprios[x] = malloc(sizeof(char) * 64);

@.*  { fprintf(output, "%s", yytext); }


    /* tags author e editor */
{T}[\ ]*[=][\ ]*["] {
    aspas = 1;
    token = strtok(yytext, " =");
    while(!strcmp(token, ""))
        token = strtok(NULL, " =");
    fprintf(output, "%s = {", token);
    otherTag = 0;
}
    /* tags author e editor */
{T}[\ ]*[=][\ ]*[{] {
    token = strtok(yytext, " =");
    while(!strcmp(token, ""))
        token = strtok(NULL, " =");
    fprintf(output, "%s = {", token);
    otherTag = 0;
}

    /* print outras tags */
[^{T}][\ ]*[=][\ ]*[{"].* {
    otherTag = 1;
    fprintf(output, "%s", yytext);
}

```

```
56      /* Final com virgula */
57  [\}\"][\,] {
58      if(aspas)
59          fprintf(output, "},");
60      else
61          fprintf(output, "%s", yytext);
62      aspas = 0;
63  }
64
65      /* Final com \n */
66  [\}\"]/[\n] {
67      if(aspas)
68          fprintf(output, "}");
69      else
70          fprintf(output, "%s", yytext);
71      aspas = 0;
72  }
73
74      /* Apelidos + nomes proprios guardados */
75  ([\ ]{N})?{LM}{L}+/([\,]|([\ ]+and|[\}\"\ ]*)) {
76      if(!otherTag) {
77          if(i == 0)
78              sprintf(apelido,"%s", yytext);
79          else
80              sprintf(apelido,"%s, ", yytext);
81
82          for(j = 0; j < i; j++)
83              strcat(apelido, nomesProprios[j]);
84
85          i = 0;
86          fprintf(output, "%s", apelido);
87          BEGIN LIMPAR;
88      }
89      else fprintf(output, "%s", yytext);
90  }
91
92      /* Guardar nomes próprios atuais */
93  [A-ZÁÉÍÓÚ]/({L}+({LM}{L}+)*([\ ]+{N}?{LM}{L}+){AUX} {
94      if(!otherTag) {
95          sprintf(nomesProprios[i], "%s.", yytext);
96          i++;
97          BEGIN LIMPAR;
98      }
99      else fprintf(output, "%s", yytext);
100 }
101
102     /* Print dos 'and' */
103 [\ \n\r]and[\ \n\r] { fprintf(output, " and "); }
104
105     /* Tratar casos de nomes do tipo: YYYY, XXXX XXXX */
106 ({LM}{L}+[\ ]?)+[\,][\ ]*{LM}{L}+([\ ]+{LM}{L}+)* {
107     if(!otherTag) {
108         token = strtok(yytext, ",");
109         nomes = strtok(NULL, ",");
110         nomes = strtok(nomes, " ");
111         fprintf(output, "%s, ", token);
112         while(nomes != NULL) {
113             int ascii = nomes[1];
114
```

```
115             //Nao é letra (<65 >122) e nao é ponto (46)
116             //Serve pra encontrar casos com acento
117             if((ascii < 65 || ascii > 122) && ascii != 46)
118                 fprintf(output, "%c%c.", nomes[0], nomes[1]);
119             else
120                 fprintf(output, "%c.", nomes[0]);
121             nomes = strtok(NULL, " ");
122         }
123     }
124     else fprintf(output, "%s", yytext);
125 }
126 
127 
128     /* Limpar restos dos matches de nomes proprios */
129 <LIMPAR>{L}|[\ ] { }
130 <LIMPAR>\n {
131     fprintf(output, "%s", yytext);
132     BEGIN(INITIAL);
133 }
134 
135     /*
136         Print o que sobrou para não se
137         perder info do ficheiro original
138     */
139 .|\n { fprintf(output, "%s", yytext); }
140 
141 
142 %%
143 
144 
145 int main (int argc, char* argv[]) {
146     FILE *input;
147     input = fopen(argv[1], "r");
148     yyin = input;
149     yylex();
150     return 0;
151 }
```

../genGraph.fl

```
1  %option noyywrap
2  
3  %{
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <string.h>
7  
8  char* multi_tok(char *input, char *delimiter);
9  %}
10 
11 L        [a-z]|ú|ù|á|à|ç|é|è|õ|ó|ò|ã|ì|í|\.
12 LM       [A-Z]|Ú|Ù|Á|À|Ç|É|É|Õ|Ó|Ò|Ã|Ì|Í
13 N        (da|dos|de)[\ ]
14 NAME     {N}?({LM}{L}+)+([\,][\ ]({LM}[\.])+)?
15 
16 %%
17 
18     FILE *output  = fopen("Graph/graph.dot", "w");
19     char **names, *token;
```

```
20      char format[64] = "\"%s\" -- \"%s\"\n";
21      int flag = 0, i, j;
22      int start = 1;
23      names = (char **) malloc(sizeof(char *) * 20);
24
25  author { flag = 1; }
26
27  {NAME}([\ ]+and[\ ]+{NAME})* {
28      if(start) {
29          fprintf(output, "strict graph G{\n");
30          fprintf(output, "ranksep=\"1.0 equally\"\n");
31          fprintf(output, "bgcolor=lightsteelblue4;\n");
32          fprintf(output, "edge [color=grey94];\n");
33          fprintf(output, "node [shape=box,style=filled,color=grey];\n");
34          start = 0;
35      }
36
37      if(flag) {
38          i = 0;
39          token = multi_tok(yytext, " and ");
40          names[i++] = strdup(token);
41          token = multi_tok(NULL, " and ");
42          while(token) {
43              j = 0;
44              while(j < i)
45                  fprintf(output, format, names[j++], token);
46              names[i++] = strdup(token);
47              token = multi_tok(NULL, " and ");
48          }
49      }
50
51      flag = 0;
52
53  }
54
55  \n|. {}
56
57  <<EOF>>   fprintf(output, "}\n"); return 0;
58
59
60  %%
61
62  char* multi_tok(char *input, char *delimiter) {
63      static char *string;
64      if (input != NULL)
65          string = input;
66
67      if (string == NULL)
68          return string;
69
70      char *end = strstr(string, delimiter);
71      if (end == NULL) {
72          char *temp = string;
73          string = NULL;
74          return temp;
75      }
76
77      char *temp = string;
78
```

```
79      *end = '\0';
80      string = end + strlen(delimiter);
81      return temp;
82  }
83
84  int main (int argc, char* argv[]) {
85      FILE *input;
86      input = fopen(argv[1], "r");
87      yyin = input;
88      yylex();
89      return 0;
90  }
```

Apêndice B

Código do Programa para Processador de Inglês corrente

../procIngles.fl

```
%option noyywrap
%{
#include <glib.h>
#include <string.h>
#include <ctype.h>
GSList* list;
GSList* iterator;
FILE *outV;
FILE *Contracoes;
%}

L            [a-z]|ú|ù|á|à|ç|é|è|õ|ó|ò|ã|ì|í
LM           [A-Z]|Ú|Ù|Á|À|Ç|É|É|Õ|Ó|Ò|Ã|Ì|Í
SU           i|you|he|she|it|we
OB           me|him|her[s]?|it[s]?|your[s]?|us|them
PP           my|his|mine|their[s]?|our[s]?
AV           how|then|more|also|here|there|still|never|when|really|back|just|by
C            for|and|nor|but|or|yet|so|if|after|before|even|in|once|now
CC           only|until|while|when|where|thought|till|whe[a-z]+
RP           that|which|who|whom|whose|whichever|who[a-z]+
A            a|an
DP           the|this|those|these
IP           any|most|none|both|many|several|anybody|anyone|anything|each
IPP          either|everbody|everyone|neither|nobody|nothing|some[a-z]+
O            not|[^a-z]|from|new|with
MV1          [Cc]an[’n]?|[Cc]ould[n’t]?|[Ss]hall|[Ss]hant|[Ss]hould[n’t]?
MV2          [Ww]ill|[Ww]on’t|[Ww]ould[n’t]?|[Mm]ay[n’t]?|[Mm]ight[n’t]?
N            1|2|3|4|5|6|7|8|9|0
NE           one|two|three|four|five|six|seven|eight|nine
AU           [Dd]id[n’t]?|[Dd]o[n’t]?|[Dd]oes[n’t]?
Q            all|some|several|enough|numerous|few|another|same
T            ([a-z]+[ijquv])|([a-z]+[rfz][fz])|([a-z]+(ing|ed|lly|[^s]s|es))
NN           [Hh]e|[Ss]he|[Ii]t
EXCEPT1      {OB}|{SU}|{A}|{O}|{N}+|{NE}([a-z]+)?|{Q}|{AV}|{Q}|{C}
EXCEPT2      {CC}|{RP}|{DP}|{PP}|{IP}|{IPP}|{T}

%%

```

```
  Contracoes = fopen("Contracoes.txt", "w");
  int j, i = 0;
  char *token;
  char* args[10];


'm                      { fprintf(Contracoes, " am"); }
're                     { fprintf(Contracoes, " are"); }
{NN}/'s                  { fprintf(Contracoes, "%s is/has", yytext); }
've                     { fprintf(Contracoes, " have"); }
'll                     { fprintf(Contracoes, " will"); }
'd                      { fprintf(Contracoes, " had/would"); }
can't                   { fprintf(Contracoes, "cannot"); }
won't                   { fprintf(Contracoes, "will not"); }
shan't                  { fprintf(Contracoes, "shall not"); }
{LM}?{L}+/n[\']t        { fprintf(Contracoes, "%s not", yytext); }

[Tt]o[\ ]({EXCEPT1}|{EXCEPT2}) {

    fprintf(Contracoes, "%s", yytext);

}

[Tt]o[\ ][a-z]+ {

    list = g_slist_insert_sorted(list,
    strdup(yytext+3),
    (GCompareFunc)g_ascii_strcasecmp);
    fprintf(Contracoes, "%s", yytext);

}


({MV1}|{MV2})[\ ]({EXCEPT1}|{EXCEPT2}) {

    fprintf(Contracoes, "%s", yytext);

}

({MV1}|{MV2})[\ ][a-z]+ {

    token     = strdup(yytext);
    args[i]   = strtok(token, " ");
    args[++i] = strtok(NULL, " ");

    list = g_slist_insert_sorted(list, args[1],
    (GCompareFunc)g_ascii_strcasecmp);
    i = 0;

    fprintf(Contracoes, "%s", yytext);

}

{AU}[\ ][a-zA-Z]+[\ ]({EXCEPT1}|{EXCEPT2}) {

    fprintf(Contracoes, "%s", yytext);

}

```

```
98
99  {AU}[\ ][a-zA-Z]+[\ ][a-z]+ {
100
101     token   = strdup(yytext);
102     args[i] = strtok(token, " ");
103
104     while(args[i] != NULL)
105         args[++i] = strtok(NULL, " ");
106
107     list = g_slist_insert_sorted(list, args[2],
108     (GCompareFunc)g_ascii_strcasecmp);
109     i = 0;
110     fprintf(Contracoes, "%s", yytext);
111
112 }
113
114 's          { }
115 n[\']t          { }
116 .|\n            { fprintf(Contracoes, "%s", yytext); }
117
118 %%
119
120
121 int main (int argc, char* argv[]) {
122     FILE *input;
123     int z = 0;
124     char *auxiliar = malloc(sizeof(char)*100), *aux;
125     char format[128], search[128], book[64];
126     char arrow[64], dict[256], pencil[64];
127     char letter, ultima = ' ';
128     strcpy(format, "<a onClick='oL(\"%c\")'><p> Letter %c%s </p></a>\n");
129     strcpy(search, "<a href='%s%s'%s><i class='fa fa-search'></i></a></li>\n");
130     strcpy(arrow, "<a><i class='fa fa-arrow-right'></i></a>");
131     strcpy(dict, "https://dictionary.cambridge.org/dictionary/english/");
132     strcpy(pencil, "<i class='fa fa-pencil'> </i>");
133     strcpy(book, "<i class='fa fa-book'></i>");
134
135     iterator = NULL;
136     list  = NULL;
137     input = fopen(argv[1], "r");
138     outV  = fopen("verbos.html","w");
139     Contracoes = fopen("Contracoes.txt", "w");
140     yyin = input;
141     yylex();
142
143     fprintf(outV, "<!DOCTYPE html> <html> <head>\n");
144     fprintf(outV, "<title> Ingles </title>\n");
145     fprintf(outV, "<link rel='stylesheet' href='css/styles.css'>\n");
146     fprintf(outV, "<link rel='stylesheet' href='css/fa/css/font-awesome.css'>\n");
147     fprintf(outV, "<script src='js/script.js'></script>\n");
148     fprintf(outV, "<meta charset='utf-8'/> </head> <body>\n");
149     fprintf(outV, "<h1 class='bg-4 title'> Verbs %s</h1>\n",book);
150
151
152     for (iterator = list; iterator; iterator = iterator->next) {
153
154         if(strcmp(auxiliar,iterator->data) != 0) {
155             aux = (char *)iterator->data;
156
```

```
157             if(ultima != aux[0]) {
158                 fprintf(outV, "</ul>");
159                 letter = toupper(aux[0]);
160                 fprintf(outV, format, letter, letter, pencil);
161                 fprintf(outV, "<ul id='%c'>\n", letter);
162                 ultima = aux[0];
163             }
164 
165             fprintf(outV, "<li> %s ", aux);
166             fprintf(outV, "%s", arrow);
167             fprintf(outV, search, dict, aux, "target='_blank'");
168         }
169 
170         auxiliar = iterator->data;
171     }
172     fprintf(outV, "</body> </html>\n");
173     return 0;
174 }
```