Biblio-Tesaurus

Daniel Costa 81302, José Fernandes 82467

Junho 2019

1 Resumo

O trabalho explicado neste relatório consiste no reconhecimento de um T2788, armazenamento deste numa estrutura e geração de um pequeno web site o com redirecionamento o qual permite a visualização do T2788.

2 Introdução

O seguinte relatório explica o terceiro trabalho da disciplina de Processamento de Linguagens.

Este trabalho envolveu a utilização do par de ferramentas Yacc/Flex para o reconhecimento e manipulação de ficheiros no formato Biblio-Tesaurus T2788.

Foi utilizado a glib-2.0 para o armazenamento intermédio do ficheiro em dos diversas estruturas de dados .

Após o reconhecimento do ficheiro foi gerado um conjunto de páginas HTML com hiperligações as quais facilitam a navegação e consulta do T2788.

3 Gramática

3.1 Terminais

'\n' , **','**

3.1.1 Tokens

STRING Representa um conjunto de carateres. **LINEBREAK** Representa as quebras de linha seguidas por espaços. É diferente de uma quebra de linha normal. **LANGUAGE** Token dos meta-dados para linguagens. **EXTERN** Token dos meta-dados para identificar relações que devem ser interpretadas como texto. **DESC** Token dos meta-dados para identificar/dar nome as relações e ou linguagens. **BASELANG** Token para identificar a linguagem base. **INV** Token para identificar relação inversa de linguagem.

3.2 Não Terminais

thesaurus Axioma. Um thesaurus pode conter metadados e conceitos. **metadados** Lista dos vários meta-dados separados por '\n'. **metadado** Representa os possiveis formatos de um meta-dado. **linguas** Lista de Línguas, serve para escrever código especifico das línguas. **conceitos** Lista dos vários conceitos separados por '\n'. **conceito** Um conceito é uma String e um conjunto de ligações separados por '\n'. **ligacoes** Uma ligação é uma String inicial a qual representa a relação seguida por uma lista de termos.
As traduções são ligações. **list** As listas de termos podem ser listas separadas de duas formas:**note** Separada por espaços. **termos** Separada por virgulas.

Em ambos os casos a lista pode ser continuada na próxima linha, separada por um **LINEBREAK**, o qual representa um '\n' seguido por espaços.

A distinção entre '\n' e LINEBREAK foi necessária porque existia a necessidade de distinguir entre os elementos que começavam logo a seguir a uma linha e os que tinham espaços a seguir ao '\n'.

3.3 GIC

```
thesaurus   : metadados  conceitos
            |
            ;

metadados   : metadado '\n' metadados
            | '\n'
            ;

metadado    : LANGUAGE linguas
            | BASELANG STRING
            | EXTERN  STRING
            | DESC STRING STRING
            | INV STRING STRING
            ;

linguas     : STRING linguas
            |
            ;

conceitos   : conceito '\n' conceitos
            | '\n' conceitos
            |
            ;
```

```
conceito    : STRING '\n' ligacoes
            ;

ligacoes    : STRING list '\n' ligacoes
            |
            ;

list        : STRING note
            | STRING termos
            ;

note        : STRING note
            | LINEBREAK STRING note
            |
            ;

termos      : ',' STRING
            | ',' LINEBREAK STRING
            | ',' STRING termos
            | ',' LINEBREAK STRING termos
            ;
```

3.4 Analisador Léxico Flex

A primeira linha do analisador léxico serve para ignorar os comentários os quais começam por # e vão até ao final da linha.

```
#.*                                 {}
```

De seguida foram definidas as palavras reservadas para identificar os tokens dos meta-dados.

```
%extern                 {return EXTERN;}
%language               {return LANGUAGE;}
%description            {return DESC;}
%baselang               {return BASELANG;}
%inv                    {return INV;}
```

De seguida foi definido a distinção entre as quebras de linha normais e as quebras de linhas seguidas de espaços.

Por motivos de comodidade ao utilizador foram consideradas todas as linhas as quais contem espaços mas não contenham qualquer conteúdo como sendo '\n'.

```
\n\ *#.*                {return '\n';}
\n\ +/[^ \n]            {return LINEBREAK;}
```

De seguida foram definidas as potencias maneiras de escrever uma string. Uma sequencia de carateres e no caso de o utilizador querer usar outros carateres pode utilizar aspas.

```
\".*?\"                 {
                            yylval.s = strdup(yytext+1);yylval.s[yyleng-2]=0;
                            return(STRING);
                        }
[A-Za-z()]+             { yylval.s = strdup(yytext); return STRING; }
```

O analizador léxico retorna os carateres '\n' e ',' e ignora espaços e tabs.

```
[\n,]                   { return yytext[0]; }
[ \t]                   {}
```

Qualquer outro carater resulta no retorno de um erro.

```
.                       { printf("%s", yytext); yyerror("Caracter inválido"); }
```

4 Estrutura Intermédia

A estrutura intermédia foi implementada utilizando a GLib.

Foram utilizadas várias estruturas para guardar as informações relativas aos meta-dados. Estas informações foram utilizadas para várias coisas durante o programa.

```
//HashSet Key String
 GHashTable *linguas;
 GHashTable *externs;
 GHashTable *description;
 GList *relacoes;
```

A hash table linguas apenas contem chaves e é utilizada para distinguir traduções de relações.

A hash table externs apenas contem chaves e é utilizada ao percorrer a estrutura para determinar como a informação deve ser imprimida.

A hash table description contem como chave o identificador da ligação e como valor uma string que a qual a descreve. Utilizada ao imprimir para alterar os nomes das ligações.

A lista relações contem os pares de ligações inversas e é utilizado na geração do website.

Os conceitos foram guardados numa árvore binária cuja chave era o termo base de forma a que a iteração sobre estes fosse feita em ordem alfabética.

```
GTree *conceitos;
```

Cada conceito foi representado por uma estrutura a qual contém a linguágem base e duas HashTables uma para as traduções e outra para as restantes ligações.

```
typedef struct conceito{
     char* termobase;
     GHashTable *traducoes;
     GHashTable *ligacoes;
}    *Conceito;
```

As hash tables têm como chave a string identificadora da ligação e como valor uma lista com todos os termos presentes na ligação.

Foram utilizadas hash tables e listas para facilitar o "merge" de listas quando as ligações se encontram separadas em 2 ou mais.

5 Modo de utilização

O programa pode ser utilizado no terminal da seguinte forma.

```
./thesaurus <exemplo.t2788
```

6 Geração do Website

As páginas HTML são geradas quando todos os dados presentes no ficheiro são guardados na estrutura de dados. Nesse momento é gerado um ficheiro chamado *index.html*. Inicialmente é gerado um cabeçalho que importa os ficheiros necessário para o bom funcionamento do site, nomeadamente *Bootstrap* e *JQuery*. Neste cabeçalho encontra-se também um pequeno *script* que permite alterar o *iframe* quando o utilizador escolhe uma palavra. Posteriormente escreve o título com um texto indicativo da língua base em que está escrito o documento.

```
void printPreIndex(FILE* f){
      fprintf(f,"<head>...</head>");
      fprintf(f,"<div class=\"jumbotron\"><h1>Biblio-Thesarus");
      if(baselang) {
            gpointer baselangName = g_hash_table_lookup(description, (gpointer) baselang);
            if(baselangName) {
                  baselang = (char*) baselangName;
            }
            fprintf(f, "<small> (%s)</small>", baselang);
      }
      fprintf(f,"</h1>...");
}
```

Posteriormente, percorre-se a lista de conceitos e adiciona-se o termo base às opções no ficheiro índice.

```
fprintf(indexF,"\n<option>%s</option>",data->termobase);
```

Para cada conceito cria-se um ficheiro e com o nome do termo base e é escrito o cabeçalho com tudo o que é necessário para o bom funcionamento da página.

```
FILE* page = fopen(filepath,"w");
fprintf(page,"<head>...</head>");
```

O título é escrito com base no termo base guardado do conceito. Posteriormente é escrito no ficheiro um segmento para traduções. São percorridos os elementos das traduções do conceito e escritas no ficheiro. Se a língua contiver uma descrição é usada a descrição da linguagem em vez da abreviatura.

```
fprintf(page,"<h1>%s</h1>",data->termobase);

g_hash_table_iter_init (&iter, data->traducoes);
fprintf(page,"<h3>Traduções</h3>\n");
while (g_hash_table_iter_next (&iter, &key, &value)) {
      char * keyName =  g_hash_table_lookup(description,key);
      if(keyName == NULL)
            keyName = (char *) key;
      fprintf(page,"<p>%s: %s</p>\n", keyName, (char *)((GList*)value)->data);
}
```

Quando é iterada todas as traduções do termo são geradas as relações inversas. Assim percorre-se o par de relações inversas e cria-se duas colunas. Em cada uma das colunas é inserido os termos de cada ligação.

```
fprintf(page,"<h3>Ligações</h3>\n");
for (GList* l = relacoes; l != NULL; l = l->next) {
      Pair keys = (Pair) l->data;
      fprintf(page,"<div class=\"row\">\n");
      fprintf(page,"<div class=\"col-6 border-right\">\n");
      printLigacao(page, keys->a1, g_hash_table_lookup(data->ligacoes, keys->a1));
      g_hash_table_remove (data->ligacoes, keys->a1);
      fprintf(page,"</div>\n");
      fprintf(page,"<div class=\"col-6 border-left\">\n");
      printLigacao(page, keys->a2, g_hash_table_lookup(data->ligacoes,keys->a2));
      g_hash_table_remove (data->ligacoes, keys->a2);
      fprintf(page,"</div>\n");
      fprintf(page,"</div>\n");
}
```

Posteriormente, são inseridos as restantes ligações. Para isso, percorre-se todas as ligações do conceito e é escrito no ficheiro do conceito.

```
g_hash_table_iter_init (&iter, data->ligacoes);
while (g_hash_table_iter_next (&iter, &key, &value)) {
      printLigacao(page, key, value);
}
```

Cada ligação contêm vários termos ou um texto. Inicialmente, é imprimido o nome da ligação. Se a abreviatura tiver descrição é imprimida a descrição em vez da abreviatura.

```
char * keyName =  g_hash_table_lookup(description,key);
if(keyName == NULL)
    keyName = (char *) key;
fprintf(page,"<h4>%s</h4>\n", keyName);
```

Se a ligação estiver marcada como *extern* ela será imprimida no ficheiro como um texto. As palavras são concatenadas com espaço e escritas no ficheiro.

```
fprintf(page,"<p>%s", (char *)((GList*)value)->data);
for (GList* l = ((GList*)value)->next; l != NULL; l = l->next) {
      fprintf(page," %s", (char*)l->data);
}
fprintf(page,"</p>\n");
```

Por outro lado, as restantes ligações são imprimidas numa lista. Os termos que tenham um conceito definido no ficheiro são realçados com uma hiperligação para o ficheiro correspondente ao termo.

```
fprintf(page, "<ul>");
for (GList* l = value; l != NULL; l = l->next) {
      if(g_tree_lookup (conceitos, l->data)) {
            fprintf(page,"<li><a href=\"%s.html\">%s</a></li>\n", (char *) l->data, (char *)
      } else {
            fprintf(page,"<li>%s</li>\n", (char *) l->data);
      }
}
fprintf(page,"</ul>\n");
```

Por fim, é imprimido no ficheiro de índice os restantes elementos na página necessários.

```
void printPostIndex(FILE* f){
      fprintf(f,"</select></div><div class=\"container\"><br/><div class=\"embed-responsive
      fclose(f);
}

g_hash_table_iter_init (&iter, data->ligacoes);
while (g_hash_table_iter_next (&iter, &key, &value)) {
      printLigacao(page, key, value);
}
```

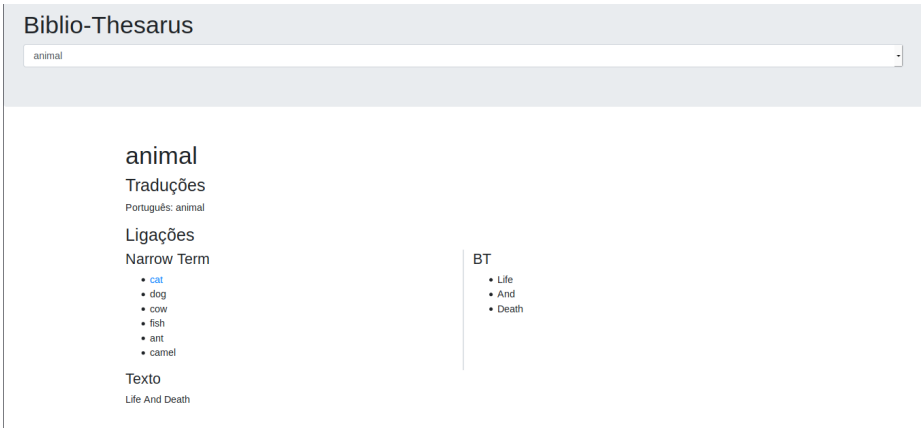

Figure 1: Página de animal.

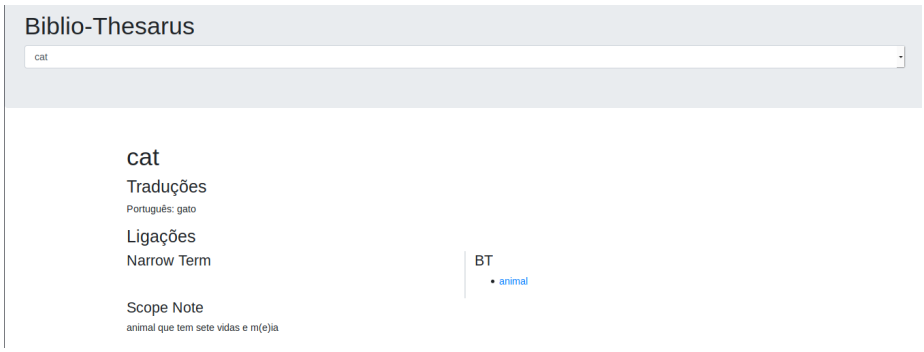

Figure 2: Página de cat.

7 Exemplo de Documento

O exemplo a seguir foi utilizado para gerar o website na geração do Website

```
%language PT EN                 # línguas: PT EN
%baselang EN                    # língua de base: EN
%inv NT BT                      # NT, BT são relações inversas a NT B => b BT a
%extern TXT                     # Declara que TXT deve ser interpretado como texto
%description TXT Texto
%description NT "Narrow Term"   # Atribuí nome para display à relação
%description SN "Scope Note"
%description PT "Português"     # Atribuir nomes também funciona nas linguas.

                                # conceitos:
animal                          # termo na baselang
PT animal                       # LINGUA termo
NT cat, dog, cow,               # NT = narrow term = termo específico
   fish, ant
BT Life And Death                       # BT = broader term = termo genérico
TXT Life And Death                       # BT = broader term = termo genérico
NT camel

                                # linha em branco : separador de conceitos

cat
PT gato
BT animal
SN animal que tem sete          # scope note = nota explicativa
   vidas e m(e)ia               # considerada texto por default

#comentário                     # desde o símbolo '#' até ao fim da linha
```

8 Conclusão

A utilização do flex em conjunto com o yacc permitiu ignorar aspetos como a utilização de espaços no programa o que permitiu o foco em aspetos mais importantes durante a especificação da linguagem.

O ignorar dos espaços no entanto causou conflitos na identificação de casos em que as linhas começam logo a seguir à mudança de linha ou contêm espaços, um aspeto importante à especificação. Este problema foi no entanto resolvido no flex e no yacc através da introdução de um token adicional o qual permite a distinção destes dois casos, o que permitiu lidar com estes casos específicos e ignorar os outros aspetos.

A utilização de uma estrutura intermédia permitiu agrupar e ordenar dados, algo que não seria possível se tivesse-mos simplesmente utilizado prints à medida que percorríamos o programa.

Um aspeto final a tomar em consideração foi a definição isolada dos meta-dados a qual permitiu a inserção de tipos de meta-dados adicionais inspirados no formato T2788.