Processamento de Linguagens (3º ano MIEI)

Trabalho Prático nº1 - Enunciado 4

Relatório de Desenvolvimento-Grupo 39

Henrique Paz (a84372), José Santos (a84288), Pedro Gomes (a84220)

5 de Abril de 2020

Resumo

O primeiro trabalho prático de Processamento de Linguagens consiste na elaboração de um projeto usando o FLEX que converte ficheiros HTML para o formato JSON. No nosso caso um ficheiro HTML de um forum de comentarios(enunciado 4). Neste relatorio vamos explicar a forma de como fizemos os filtros FLEX, que nos permitiram recolher a informação do ficheiro HTML, a estrutura que usamos para guardar os dados e tambem a escrita de cada registo para o formato JSON. Os nossos principais desafios neste trabalho foram desenvolver as nossas competencias com o FLEX e a utlização de expressoes regulares.

Conteúdo

Capítulo 1

Introdução

1.1 Contexto

O Publico_extraction_portuguese_comments_4.html contem centenas de comentários de diferentes utilizadores. Cada comentário tem tag de ID do comentário, o nome do utilizador, a data e hora em que o comentário foi feito e o texto do comentário. O objetivo deste projeto consiste em extrair a informação do ficheiro html e, organiza-la e depois apresenta-la no formato JSON.

1.2 Desafios

Na primeira fase o desafio passa por através do flex tirar toda a informação do ficheiro e guardar essa informação numa estrutura de dados definida por nos. De seguida conferimos que estava a ser guardada bem a informação e por ultimo imprimi-la para o formato JSON.

1.3 Opçoes tomadas

Uma vez que existem inúmeros comentários no ficheiro HTML a analisar decidimos usar a biblioteca Glib de modo a conseguirmos de forma mais eficaz fazer a nossa estrutura de dados, com os tipos de dados apropriados. Também decidimos ter uma estrutura principal(CommentThread head) com todos os comentários no forum e cada comentário tem na sua estrutura as suas respostas, esta estrutura é iniciada no filtro.l e é a mesma que é utilizada para escrever no formato json visto que tem toda a informação que extraimos.

Capítulo 2

Conceção/Desenho da Resolução

2.1 Estrutura de dados

Para armazenar a informação que é retirada pelo filtro flex decidimos contruir uma estrutra de dados que nos permite guardar o ID do comentário, o nome do utilizador,a data e hora que o comentário foi feito, o texto do comentário e todas as respostas a esse comentário.

```
typedef struct commentThread {

    GString * id;
    GString * user;
    GString * date;
    GString * timestamp;
    GString * comentTxt;
    int hasReplies;
    int numberOfReplies;
    CommentThread replies[];
}*CommentThread;
```

Figura 2.1: Struct commentThread

No caso do id,nome,data,hora e texto decidimos usar strings,o numero de respostas é um inteiro e se o comentário tem repostas só pode ser TRUE ou FALSE.Por fim temos as replies que é um array de commentThread, para podermos recursivamente percorrer a estrutura toda. Tudo isto foi definido no commentThread.c

2.2 Implementação

Tal como foi mencionado anteriormente o filtro extrai a informação do ficheiro HTML e para isso é usado um filtro FLEX para tal.

Primeiro começamos por converter o ficheiro imput para UTF8 e demos o nome de Publico-utft8.html. De seguida quando começamos a analisar o ficheiro notamos que existe uma tag,<ol class="comments___list"id="approved-comments", que aparece unicamente no inicio do ficheiro e por isso usamo-la para iniciar a estrutura "head"que vai conter todos os comentários que estão na pagina.

```
\<(ol)[' ']{class}\=\"{comments__list}\"[' ']{id}\=\"{approved}\-{comments}\"\> {head = newCommentThread();}
```

Figura 2.2: Tag que inicia a estrutura.

De seguida já sabemos que temos a nossa estrutura principal iniciada e podemos começar a procurar pelos comentários. No caso do ID do comentário sabemos que acontece sempre que encontramos uma <li class="comment">, o nome do utilizador é quando aparece uma <h5 class="comment_author">, a data e hora é a tag <time class, o texto do comentário é a tag , e quando esse comentário tem respostas aparece a tag <ol class="comments_list">e que até aparecer o feixo da tag tem todas as respostas ao comentário.

Visto isto usamos start conditions(SC) para tratar da informação uma vez apanhada a tag.

ID USER TIMESTAMP TEXT foram as SC que nos usamos.

No caso da SC ID, quando iniciada, a ideia por detras foi rejeitar tudo o que não era o ID, o igual, o texto desnecessario(data-comment-id),o newline entre outas coisas e aceitar tudo o resto que seria o id do comentário e nesse caso adicionar a estrutura commentThread.

```
\<(li)[' ']{class}\=\"{comment}\"      {curr = addnewComment(giveThread());BEGIN ID;}
<ID>\>                                 {BEGIN INITIAL;}
<ID>[' ']                              {;}
<ID>\"                                 {;}
<ID>\n                                 {;}
<ID>\=                                 {;}
<ID>data\-comment\-id                  {;}
<ID>.                                  {setID(curr,yytext);}
```

Figura 2.3: Start condition ID.

A SC USER, quando iniciada, trata de filtar o nome do utlizador que criou o comentário e teve a mesma logica da SC ID visto que o nome de um utilizador não tem um formato uniforme logo para nós pareceu mais facil dizer o que não aceitavamos e tudo o resto consideravamos como nome do utilizador.

```
\<(h5)[' '](class)\=\"(comment__author)\"\>          {BEGIN USER;}
<USER>\<\/h5\>                                      {BEGIN INITIAL;}
<USER>\n                                            {;}
<USER>[' '][' '][' ']+                              {;}
<USER>\<(a)[' '](href)\=\"(.*)\"[' '](rel)\=\"(.*)\"\>  {;}
<USER>\<\/a\>                                       {;}
<USER>.                                             {setUser(curr,yytext);}
```

Figura 2.4: Start condition USER.

No caso da SC TIMESTAMP foi diferente visto que verificamos que tinha sempre a mesmas estrutura então procuramos especificamente por 2 numeros seguido de um ponto mais 2 números um ponto e mais 4 números para a data e no caso da hora procuramos por dois numeros seguidos de 2 pontos e de novo 2 números.

```
\<(time)[' '](class)\=\"                    {BEGIN TIMESTAMP;}
<TIMESTAMP>\<\/time\>                       {BEGIN INITIAL;}
<TIMESTAMP>[0-9]{2}\:[0-9]{2}\:[0-9]{2}\.[0-9]+  {;}
<TIMESTAMP>[0-9]{2}\.[0-9]{2}\.[0-9]{4}     {setDate(curr,yytext);}
<TIMESTAMP>[0-9]{2}\:[0-9]{2}               {setTimeStamp(curr,yytext);}
<TIMESTAMP>.|\n                             {;}
```

Figura 2.5: Start condition TIMESTAMP.

A SC TEXT serve para filtrar o texto do comentário e tal como para o ID e para o USER rejeitamos o newline e as aspas e tudo o resto assumimos como texto e é adicionado a estrutura.

```
\<(div)[' '](class)\=\"(comment__content)\"\>  {BEGIN TEXT;}
<TEXT>\<\/p\>                                  {curr = getCurrentReply(head);BEGIN INITIAL;}
<TEXT>\<p\>                                    {;}
<TEXT>\n                                       {;}
<TEXT>\"                                       {;}
<TEXT>[' '][' '][' ']+                         {;}
<TEXT>.                                        {addCommentTxt(curr,yytext);}
```

Figura 2.6: Start condition TEXT.

Para guardar as respostas ao comentario reparamos que era iniciada uma tag como foi dito anteriormente e so no fim de todas as respostas ao comentario é que era fechada.Visto isto decidimos usar uma flag que diz se o comentario que estamos a encotrar é uma resposta a outro ou não. Devido a isto decidimos tambem ter um apontador para o comentario que estamos neste momento a trabalhar (CommentThread curr).Deste modo quando encontramos um comemtario que nao é responda, ele é simplesmente adicionado a head, quando esse comentario que encontramos é uma resposta ele é adicionado ao curr e o apontador do curr é atualizado. É para isso que serve a funçao giveThread() que é definida no inicio do filtro flex, que conforme o valor que a variavel isreply tem dá return a commentThreads diferentes.A variavel isreply é igual a zero quando não é uma resposta e igual a um quando se trata de uma resposta a um comentario.

```
\<(ol)[' ](class)\=\"(comments__list)\"\>                    {isreply = 1;}
\<\/ol\>                                                     {isreply = 0;curr = getCurrentReply(head);}
```

Figura 2.7: Inicio e fecho da tag de respostas

Na funçao main é criado o ficheiro de output em json com o nome que lhe é passado com a funçao openFile() e de seguida é carregada para la toda a informaçao toda com a funçao formatToJsonHead() passando a head que é a estrutura que contem toda a informaçao que extraimos do ficheiro.

```
openFile("output.json");

formatToJsonHead(head);
```

Figura 2.8: funçao main

2.3 Makefile

Para a compilaçao do programa criamos uma makefile, para deste modo ser mais facil correr o programa sendo apenas necessario escrever make no terminal na diretoria em que a makefile está inserida.

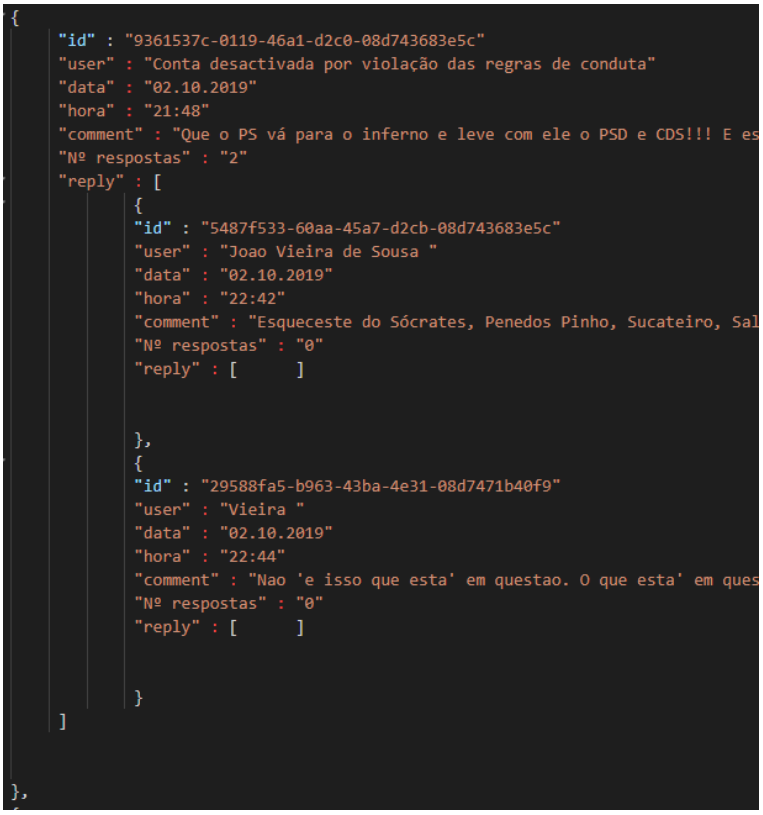
```
./filtro < Publico-utft8.html
```

Figura 2.9: Exec do programa

Capítulo 3

Testes

Exemplo de parte do ficheiro output no formato JSON resultado de um teste efetuado tendo como imput o ficheiro Publico-utft8.html.



```
{
    "id" : "9361537c-0119-46a1-d2c0-08d743683e5c"
    "user" : "Conta desactivada por violação das regras de conduta"
    "data" : "02.10.2019"
    "hora" : "21:48"
    "comment" : "Que o PS vá para o inferno e leve com ele o PSD e CDS!!! E es
    "Nº respostas" : "2"
    "reply" : [
            {
            "id" : "5487f533-60aa-45a7-d2cb-08d743683e5c"
            "user" : "Joao Vieira de Sousa "
            "data" : "02.10.2019"
            "hora" : "22:42"
            "comment" : "Esqueceste do Sócrates, Penedos Pinho, Sucateiro, Sal
            "Nº respostas" : "0"
            "reply" : [      ]


            },
            {
            "id" : "29588fa5-b963-43ba-4e31-08d7471b40f9"
            "user" : "Vieira "
            "data" : "02.10.2019"
            "hora" : "22:44"
            "comment" : "Nao 'e isso que esta' em questao. O que esta' em ques
            "Nº respostas" : "0"
            "reply" : [      ]


            }
    ]


},
```


Figura 3.1: Exec do programa

Capítulo 4

Conclusão

Através do desenvolvimento deste projeto conseguimos melhorar as nossas competencias relativamente a construçao de filtros Flex, de expressoes regulares e de start conditions.Os filtros flex provaram se realmente muito eficazes na leitura do ficheiro em questao, uma vez que conseguimos ler um ficheiro com inumeros comentarios e converter essa informaçao para json. Isto mostra que a utlizacao de filtros flex faz realmente muita diferença principalmente quando o objetivo é recolher informação e eventualemnte trata-la.

Para concluir achamos que fizemos um trabalho satisfatório e temos a certeza que as nossas competencias em flex ficaram claramente reforçadas para um futuro proximo, o que nos deixa muito satizfeitos visto que temos a certeza que é uma competencia que nos vai acompanhar ao longo do curso e eventualmente carreira.