Universidade do Minho

Departamento de Informática

Processing an Angolan Newspaper

Pedro Mendes (a79003)

July 8, 2019

Abstract

This project's goal is to process a million line long file to produce a comprehensive and organized collection of HTML files, adopting *flex* to generate an efficient parser using regular expressions.

Contents

Chapter 1

Introduction

This project aims to parse a very large file with articles from a newspaper in order to organize them in individual articles, indexing them by title or tag.

The main tool used for the parser is *flex*, (and the C programming language) together with the GLib library, to produce an efficient parser. On top of this a *bash* script was written to pre-process the input file.

First we'll analyse the problem, see what needs to be implemented and what challenges need to be overcome to implement said features.

Next we'll look at the solution, split into the parser's structure, how it was split into different sub contexts and what regular expressions were used to build it, and the project's architecture where we take a brief look into the data structures used.

Finally a brief analysis of the programs performance will be presented, alongside an interesting finding about performance of regular expressions.

Chapter 2

Problem

The program must fulfil the following requirements:

- Split the input file into different smaller files, one for each article, in a correct HTML DOM tree, as to be read by any ordinary web browser;
- Create an index for all the generated files to facilitate access to each of them;
- Count the occurrence of each tag and produce a comprehensive summary of them;
- Create a tag $\rightarrow$ article association;
- Create an article $\rightarrow$ tag association.

The input file presents a few challenges that need to be addressed, in order to fulfil the proposed requirements.

The first and most obvious one is the file's size. The file is millions of lines long which means that the information parsed needs to be flushed as soon as it's not needed anymore, as to not risk allocating too much memory.

The second is the publication's format, which in some cases does not lend itself to clean regular expressions.

```
<pub>
#TAG: tag:{Adeptos} tag:{Alexandre Grasseli} tag:{Girabola} tag:{Petro de Luanda}
#ID:{post-1090 post type-post status-publish format-standard has-post-thumbnail hentry category-desport
o tag-adeptos tag-alexandre-grasseli tag-girabola tag-petro-de-luanda}
Desporto

Adeptos do Petro contestam Alexandre Grasseli

PARTILHE VIA:
#DATE: [116eb] Redacção F8 – 22 de Setembro de 2014
[aa18]
A derrota do Petro de Luanda diante do 1º de Maio por 1-0, na última jornada do Girabola, levou à
contestação do treinador Alexandre Grasseli por parte dos adeptos “petrolíferos”.

A derrota em casa diante do 1º de Maio levou muitos adeptos do Petro de Luanda a manifestarem-se
contra a continuidade de Alexandre Grasseli. Foram arremessados objectos para dentro de campo e
ouviram-se assobios contra o treinador dos actuais sétimos classificados no Girabola.

Em declarações à Bola Angola, o treinador desvalorizou a contestação dos adeptos.

“O Petro é um clube grande e vencedor. Esta é uma altura para nos unirmos, pois só assim é que a
vitória pode chegar”, afirmou Alexandre Grasseli.

Etiquetas: AdeptosAlexandre GrasseliGirabolaPetro de Luanda
</pub>
```

Figure 2.1: Example Publication

As can be seen in the example above (Figure 2.1), a publication is split in roughly 2 parts, the header, in red, where the post's metadata is stored, and the body or text of the publication, in green. The first area has tags, id and date which are easy to find due to their `#NAME{` syntax, but the category and the title (in this example *"Desporto"* and *"Adpetos de Petro contestam Alexandre Grasseli"* respectively) have to be parsed using the rest of the header as context. Next, sometimes there is a sequence delimited by square brackets before the text, this is also intended to be eliminated.

The third problem to be addressed is the fact that most publications are repeated throughout the file which can interfere with the counting of the number of occurrences of each tag.

Lastly, the fourth problem is that *flex* is not unicode aware, which means that it reads one byte at a time, and since the first byte of the horizontal bar is the same as the first byte of many other unicode characters, they are indistinguishable from *flex*'s perspective. To solve this a simple *bash* script (`clean_unicode.sh`) was written to replace the horizontal bar with a sequence of `#`.

Chapter 3

Solution

3.1 Parsing contexts

To parse each publication 4 subcontexts were implemented: *HEADER* (in red), *DATE* (in blue), *TEXTHEADER* (in yellow) and *TEXT* (in green) (See Figure 3.1), zones in white are ignored.

```
<pub>
#TAG: tag:{Adeptos} tag:{Alexandre Grasseli} tag:{Girabola} tag:{Petro de Luanda}
#ID:{post-1090 post type-post status-publish format-standard has-post-thumbnail hentry category-desport
o tag-adeptos tag-alexandre-grasseli tag-girabola tag-petro-de-luanda}
Desporto

Adeptos do Petro contestam Alexandre Grasseli

PARTILHE VIA:
#DATE: [116eb] Redacção F8 – 22 de Setembro de 2014
[aa18]

A derrota do Petro de Luanda diante do 1º de Maio por 1-0, na última jornada do Girabola, levou à
contestação do treinador Alexandre Grasseli por parte dos adeptos “petrolíferos”.

A derrota em casa diante do 1º de Maio levou muitos adeptos do Petro de Luanda a manifestarem-se
contra a continuidade de Alexandre Grasseli. Foram arremessados objectos para dentro de campo e
ouviram-se assobios contra o treinador dos actuais sétimos classificados no Girabola.

Em declarações à Bola Angola, o treinador desvalorizou a contestação dos adeptos.

“O Petro é um clube grande e vencedor. Esta é uma altura para nos unirmos, pois só assim é que a
vitória pode chegar”, afirmou Alexandre Grasseli.

Etiquetas: AdeptosAlexandre GrasseliGirabolaPetro de Luanda
</pub>
```

Figure 3.1: Publication sections breakdown

3.1.1 HEADER

In the header, the id, tags, category and title are parsed. The tags are relatively easy to parse, the same goes for the id. To parse the category and title, a variable is used to check if the category has been parsed, since the same regular expression is used for both.

3.1.2 DATE

The date context is started when the horizontal bar is found, and all input strings are ignored until the `#DATE` string is matched. At this point the date is stored and the next context is started.

3.1.3 TEXTHEADER

This context serves only to remove the text between square brackets at the top of the body, then immediately switches to the `TEXT` context.

3.1.4 TEXT

This context is very simple, it simply writes what it finds to the output file, ignoring lines starting with *Etiquetas:* and stopping when it finds the end of the publication.

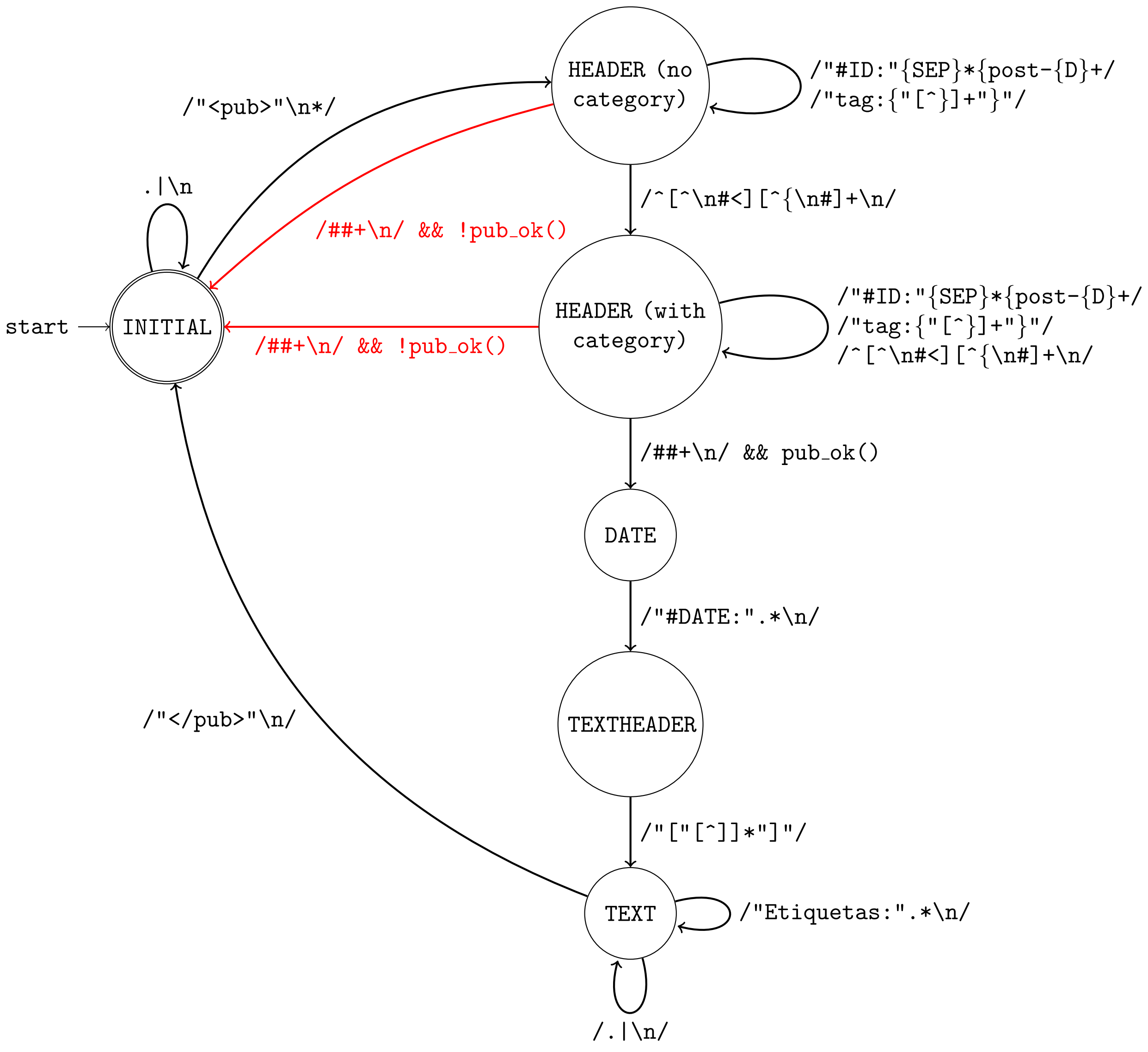


Figure 3.2: Parser state machine

As shown in Figure 3.2, there are two error paths that can be taken from the *HEADER* context back to the *INITIAL* context in case the end of the header (marked by the sequence of #) is reached and no 'id' has been parsed.

3.2 Project Architecture

Handling of the parsed information is split in two modules, `publication` and `newspaper`.

3.2.1 Publication

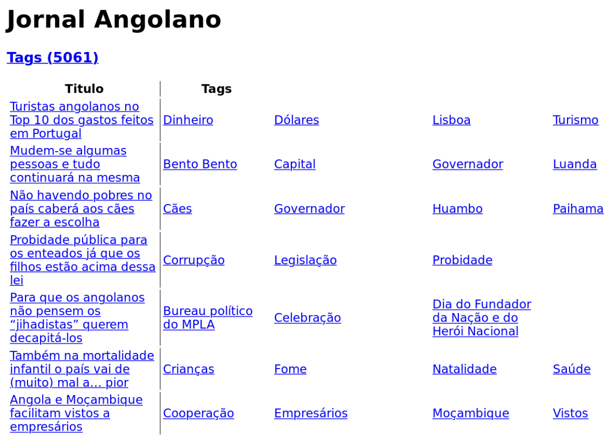
This module gathers the information of a single publication to create the corresponding `post-id.html` file. When a publication is found this module is initialized and, as the parser runs, this structure accumulates the `id`, `title`, `author_date`, `category` and `tags`. After the header is flushed to its file, every sub string parsed of the body of the publication is immediately flushed to the file, as to not accumulate too many bytes in memory.

3.2.2 Newspaper

This module indexes posts and their tags to create several files.

- `index.html`: which references all the parsed posts in a list next to their tags;
- `tags.html`: which lists all the tags found next to their occurrence count;
- `tagname.hmtl`: for every tag which lists every file with that tag.

To achieve this, two tables are kept in memory during the parsing of the whole input string, one associates post ids with the post's title and tags, the other associates tags with their posts. This way, the relations shown below can be achieved.

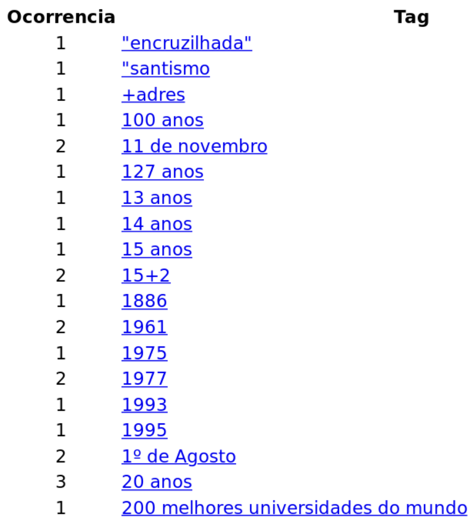


Jornal Angolano

Tags (5061)

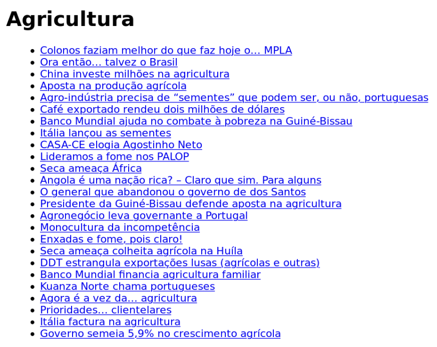
Titulo	Tags			
Turistas angolanos no Top 10 dos gastos feitos em Portugal	Dinheiro	Dólares	Lisboa	Turismo
Mudem-se algumas pessoas e tudo continuará na mesma	Bento Bento	Capital	Governador	Luanda
Não havendo pobres no país caberá aos cães fazer a escolha	Cães	Governador	Huambo	Paihama
Probidade pública para os enteados já que os filhos estão acima dessa lei	Corrupção	Legislação	Probidade	
Para que os angolanos não pensem os "jihadistas" querem decapitá-los	Bureau político do MPLA	Celebração	Dia do Fundador da Nação e do Herói Nacional	
Também na mortalidade infantil o país vai de (muito) mal a... pior	Crianças	Fome	Natalidade	Saúde
Angola e Moçambique facilitam vistos a empresários	Cooperação	Empresários	Moçambique	Vistos

(a) index.html



Ocorrencia	Tag
1	"encruzilhada"
1	"santismo
1	+adres
1	100 anos
2	11 de novembro
1	127 anos
1	13 anos
1	14 anos
1	15 anos
2	15+2
1	1886
2	1961
1	1975
2	1977
1	1993
1	1995
2	1º de Agosto
3	20 anos
1	200 melhores universidades do mundo

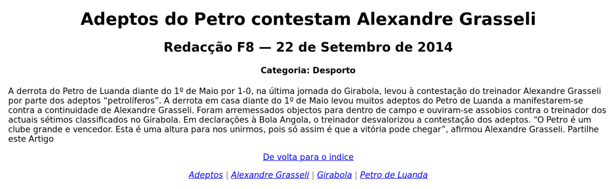
(b) tags.html



Agricultura

- Colonos faziam melhor do que faz hoje o... MPLA
- Ora então... talvez o Brasil
- China investe milhões na agricultura
- Aposta na produção agrícola
- Agro-indústria precisa de "sementes" que podem ser, ou não, portuguesas
- Café exportado rendeu dois milhões de dólares
- Banco Mundial ajuda no combate à pobreza na Guiné-Bissau
- Itália lançou as sementes
- CASA-CE elogia Agostinho Neto
- Lideramos a fome nos PALOP
- Seca ameaça África
- Angola é uma nação rica? – Claro que sim. Para alguns
- O general que abandonou o governo de dos Santos
- Presidente da Guiné-Bissau defende aposta na agricultura
- Agronegócio leva governante a Portugal
- Monocultura da incompetência
- Enxadas e fome, pois claro!
- Seca ameaça colheita agrícola na Huíla
- DDT estrangula exportações lusas (agrícolas e outras)
- Banco Mundial financia agricultura familiar
- Kuanza Norte chama portugueses
- Agora é a vez da... agricultura
- Prioridades... clientelares
- Itália factura na agricultura
- Governo semeia 5,9% no crescimento agrícola

(c) tag.html



Adeptos do Petro contestam Alexandre Grasseli

Redacção F8 — 22 de Setembro de 2014

Categoria: Desporto

A derrota do Petro de Luanda diante do 1º de Maio por 1-0, na última jornada do Girabola, levou à contestação do treinador Alexandre Grasseli por parte dos adeptos "petrolíferos". A derrota em casa diante do 1º de Maio levou muitos adeptos do Petro de Luanda a manifestarem-se contra a continuidade de Alexandre Grasseli. Foram arremessados objectos para dentro de campo e ouviram-se assobios contra o treinador dos actuais sétimos classificados no Girabola. Em declarações à Bola Angola, o treinador desvalorizou a contestação dos adeptos. "O Petro é um clube grande e vencedor. Esta é uma altura para nos unirmos, pois só assim é que a vitória pode chegar", afirmou Alexandre Grasseli. Partilhe este Artigo

De volta para o indice

Adeptos | *Alexandre Grasseli* | *Girabola* | *Petro de Luanda*

(d) publication.html

Figure 3.3: Files generated by the newspaper module

Chapter 4

Performance

Program performance was a concern during it's implementation. As such, for every major feature or change made to the program, time and memory benchmarks were ran. With this system it was possible to detect a huge performance increase when a simple change of regular expression was made.

One of the regular expressions initially used was incorrect and exceedingly slow.

```
AN   [0-9A-Za-zÀ-ÖØ-öø-ÿ\-]
ANS  [0-9A-Za-zÀ-ÖØ-öø-ÿ\- ]

<HEADER>{AN}{ANS}+\n
```

Figure 4.1: First regular expression to capture the category and title

This was made in an effort to capture accented characters, but because, once again, *flex* is not unicode aware, these ranges did not work properly. To address this, a different strategy was used: instead of listing what characters were to be matched, we listed what characters were not to be matched.

```
<HEADER>^[^\n#<][^{\n#]+\n
```

Figure 4.2: Correct regular expression to capture the category and title

This change cut 40% of the program's runtime, (benchmarked from 2.5s to 1.5s) because the list of bytes that had to be compared with was smaller, with the added benefit of capturing more precisely titles that had unicode characters, such as “, ”, –, etc.

The benchmarks made during the software's development can be seen in Apendix B.

Chapter 5

Testing

To test the correctness of the program, a few bash utils were used, notably common *unix* programs like *grep*, *cat*, *sed*, *sort*, *uniq* and, of course, the *bash*.

5.1 Number of articles

To obtain the number of unique articles in the input file the following command was used:

```
grep -Po 'post-[0-9]+' input/folha8.OUT.txt \
    | sort \
    | uniq -c \
    | sort -n -r \
    | cut -d' ' -f7 \
    | uniq -c
```

Figure 5.1: Count the amount of article repetitions and count the non repeating articles.

This command outputs the total amount of times articles are repeated, for example, for the input file tested, the output was the following:

```
3579 2
1610 1
```

Meaning that 3579 articles are repeated 2 times and 1610 articles aren't repeated. Totalling to 5189 articles. To check that the program was producing the correct amount of articles the command `ls noticias/post*.html | wc -l` was used.

5.2 Counts of given tag

The next test run was: obtaining the count of individual tags to see if they match, both in the produced articles, and `tags.html` file.

To achieve this two commands were used. (As an example the *'regime'* tag was used).

```
grep -oP 'tags/regime.html' noticias/post*.html | wc -l
```

Figure 5.2: Count the number of times the 'regime' tag comes up in the processed articles

```
grep 'tags/regime.html' noticias/tags.html \
    | awk -F'>' '{print $3}' \
    | grep -oP '[0-9]+'
```

Figure 5.3: Show the occurrences of the ‘regime’ tag comes up in the tags.html file

5.3 Count the number of unique tags

The final test ran was counting the number of unique tags in the input file to see if it matched the number on the `index.html` file. To achieve this the command on figure 5.4 was used.

```
tr '\n' ' ' < input/folha8.OUT.txt \
    |  grep -oP 'tag\:\{[^}]+' \
    | sed -E 's|tag\:\{([^}]+)|\1|' \
    | sort \
    | uniq \
    | wc -l
```

Figure 5.4: Count the number of unique tags in the input file

Chapter 6

Conclusion

In conclusion, all the requirements set out in the assignment were fulfilled, as well as some extra functionality. The usage of regular expressions to specify how certain input strings should be handled proved to be very powerful and time saving.

As an extension of this work a *CSS* file could be added to improve the visual appeal of the generated pages.

Appendix A

Flex

```
%%
"<pub>"\n*                      { BEGIN HEADER; pub_init(); has_category = 0; }
<HEADER>"tag:{"[^}]+"}"         { yytext[yyleng - 1] = '\0'; pub_tags_append(yytext + 5); }
<HEADER>"#ID:"{SEP}*\{post-{D}+ {
                                    char* id = yytext;
                                    while(*(id++) != '{'); // skip to 'post'
                                    pub_id_add(id);
                                }
<HEADER>^[^\n#<][^{\n#]+\n      {
                                    yytext[yyleng - 1] = '\0';
                                    has_category++ ?
                                    pub_title_append(yytext)
                                    : pub_category_add(yytext);
                                }
<HEADER>##+\n                   { if(pub_ok()) { BEGIN DATE; } else { END_PUB; }}
<HEADER>.|\n                    { ; }
<DATE>"#DATE:".*\n              {
                                    yytext[yyleng - 1] = '\0';
                                    char* date = yytext + strlen("#DATE:");
                                    while(*(date++) != ']'); // skip to after ']'
                                    while(*(++date) == ' '); // skip spaces
                                    pub_author_date_add(date);
                                    if(!pub_header_print()) { END_PUB; }
                                    BEGIN TEXTHEADER;
                                }
<TEXTHEADER>"["[^]]*"]"         { BEGIN TEXT; }
<TEXTHEADER>.|\n                { pub_append_text(yytext); BEGIN TEXT; }
<TEXT>.|\n                      { pub_append_text(yytext); }
<TEXT>"Etiquetas:".*\n          { ; }
<TEXT>"</pub>"\n                {
                                    pub_footer_print();
                                    END_PUB;
                                }
<*>"PARTILHE VIA:"{SEP}*        { ; }
<INITIAL,DATE>.|\n              { ; }
%%
```

Appendix B

Benchmarks

Program Version	mean (s)	stddev (s)	heap (Mb)		stack peak (Kb)
			total	peak	
Not checking for repeated articles	2.5803	0.0536	44.220	0.690	4.800
Add Article → Tag Relation	2.5413	0.0509	45.390	1.199	5.008
Add Index html file	2.5771	0.0460	45.672	1.199	5.024
Refactor Regex (Figure 4.2)	1.5959	0.0583	44.674	1.135	5.008
Add Tag → Article Relation	2.0040	0.2136	63.731	1.060	4.976
Join both Relations	1.9717	0.1553	69.157	1.848	4.720

Table B.1: The benchmarks taken during the software's development