Processamento de Linguagens e Compiladores(3º ano de LCC)

Trabalho Prático 4

Relatório de Desenvolvimento

Bruno Jardim	Eduardo Freitas	Pedro Fernandes
(A91680)	(A91643)	(A91699)

17 de novembro de 2021

Resumo do Trabalho Prático

Para este trabalho prático foi-nos proposto a filtração de um documento BibTEX, que é uma ferramenta de formatação de citações bibliográficas em documento LATEX. O formato usado separa as informações conforme pedido no enunciado ou seja, filtra toda a informação não necessária e apresenta a tag e a chave que representam o texto filtrado.Assim sendo, também fornece ao utilizador o autor e o título de todas as obras filtradas. Para além desta informação, é também apresentado um índice de autores com as chaves que representam todas as suas obras.

Conteúdo

Capítulo 1

Introdução

Neste documento, o objetivo é analisar de que forma e que ferramentas o nosso grupo utilizou no tratamento e filtragem de informação de texto de um documento BibTEX.

1.1 Um Processador de BibTEX

Área: Processamento de Linguagens

Enquadramento: Processamento de texto

Problema 1: Pretendemos filtrar as categorias e fazer a contagem das mesmas.

Problema 2: Complementar o processador anteriormente criado de forma a apresentar os autores e título das obras.

Problema 3: Criar um índice com os autores e as *tags* únicas dos textos escritos pelo mesmo.

Problema 4: Construir um grafo com as relações dos autores

Objetivo: O objetivo do relatório é demonstrar a nossa abordagem e as decisões tomadas para a resolução do problema em questão, explicando e justificando cada passo dado.

Estrutura do Documento:

1º Capítulo: Introdução.
2º Capítulo: explicaremos que tipo de informação nos foi fornecida no enunciado.
3º Capítulo: expomos como abordamos os problemas propostos, bem como que escolhas e decisões foram tomadas na sua resolução.
4º Capítulo: demonstraremos a partir de um exemplo de que forma a informação será filtrada.
5º Capítulo: Conclusão.
6º Capítulo: Apêndice - Código do programa.

Capítulo 2

Informação a tratar

Neste capítulo iremos explorar um exemplo para demonstrar o tipo de informação que o nosso processador consegue tratar. Num documento BibTEX temos um *bloco* que começa por uma *tag* que começa por *"@"* seguida de uma palavra, depois terá uma *chave* única que nos permite facilmente encontrar uma obra específica, de seguida terá informação da obra. Com esta informação iremos preceder a uma contagem de ocorrências de cada *tag*, algo que de seguida é colocado num documento HTML.

Capítulo 3

Filtragem e tratamento da informação

Neste capítulo analisaremos o desenvolvimento do nosso programa bem como abordaremos as decisões que tivemos de tomar na realização do mesmo. Sendo assim, vamos explicar como é que a informação é filtrada e ainda como é realizado o armazenamento da mesma. O primeiro processo de filtramento é detetar um *bloco*, após a deteção teremos que filtras as diversas informações que necessitamos.

3.1 Alínea a) : Análise e contagem de tags.

Para este efeito usamos uma ER para detetar quando encontramos uma tag e usamos a função *re.split()* da biblioteca *re*, para obtermos a tag não formatada. Formatamos as tags a partir da função *.title()* do Python para não haver variações no formato e efetuamos uma contagem das ocorrências.

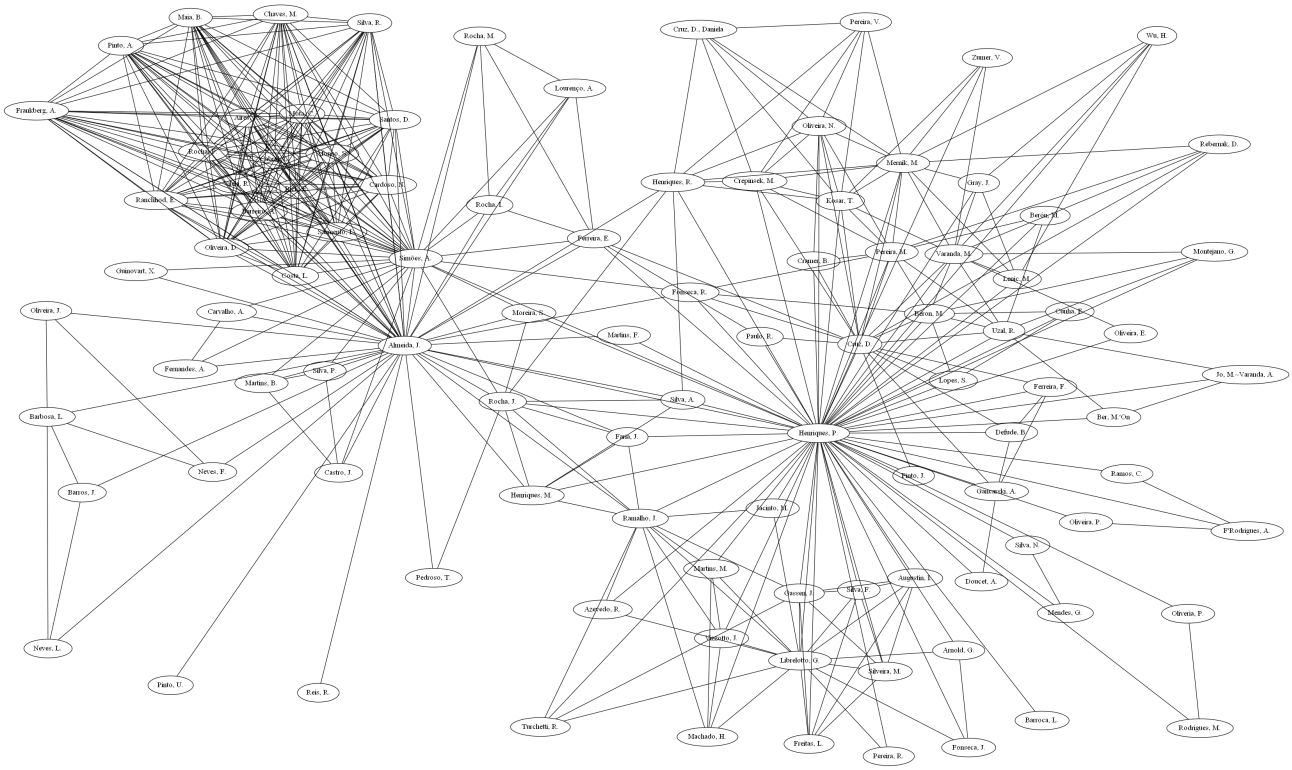
3.2 Alínea b) : Filtragem de chaves, autores e títulos.

A partir da mesma ER referida em 3.1 usamos a segunda parte do *re.split()* para obtermos a chave única de cada obra.
De seguida procedemos a filtrar a informação necessária (autor/autores e título), e.g., *author* = {*Daniela da Cruz and Pedro Rangel Henriques*}, seguidamente adicionamos a informação a um dicionário estruturado da seguinte forma {*'tag'*{*'chave'*:{{*'author'*:*'nome do autor'*},{*'title'*:*'titulo texto'*}}}

3.3 Alínea c) : Índice de autores.

Para a resolução deste exercício optamos pela criação de um dicionário que associa os autores às respetivas chaves, através da reutilização do dicionário anteriormente criado na alínea b). Neste caso, este novo dicionário possui os nomes ainda não formatados da maneira que pretendia-mos, ou seja, para além de ainda não estarem individualizados também ainda se encontram no formato original. Desta forma, para realizarmos a formatação dos nomes foi feito um *re.split()* nos "And", obtendo assim uma lista com todos os nomes desagregados. De seguida, os nomes foram filtrados de modo a apresentarem a formatação, e.g., *"Henriques, P."*.
Para concluir, criamos um dicionário onde as *keys* são os autores com os nomes formatados e os *values* uma lista com todas as chaves dos textos em que o autor esteve envolvido.

3.4 Alínea d) : Grafo de relacionamento entre autores.



Para ser possível criar este grafo usamos o dicionário utilizado em 3.3.
O processo para estabelecer as relações é o seguinte:

- Percorrer os autores todos.
- Para cada autor, percorrer todas as chaves.
- Percorrer todos os autores e verificar se eles têm a mesma chave.
- Caso sim criamos um *tuplo* com o nome dos dois autores e armazena-mo-lo numa lista.
- Finalmente passamos os *tuplos* ao *PyGraphViz* e o grafo é construído.

Capítulo 4

Apresentação da Informação Filtrada

Neste capítulo, vamos demonstrar com a ajuda de um exemplo a utilização do nosso programa. Assim sendo, utilizaremos um bloco aleatório para comprovar as formatações realizadas pelo nosso processador.

```
   @Book{RH02,
     author=   {José Carlos Ramalho and Pedro Rangel  Henriques},
     title =   {XML \& XSL: da teoria à prática},
     publisher =   {Editora FCA},
     series =   {Série Tecnologias de Informação ISBN-972-722-347-8},
     edition=  {1st Ed.},
     month =   {Oct},
     year  =   {2002}
}
```

Através dos processos descritos no Capítulo 3 quando tratamos a informação obtemos o seguinte output no código HTML.

```
@Book:1
              RH02
                            Author: José Carlos Ramalho And Pedro Rangel Henriques
                            Title: Xml \& Xsl: Da Teoria À Prática
```

Através dos processo descritos na alínea 3.3 obtemos o seguinte output, como podemos observar a chave *RH02* está presente no índice do autor *Ramalho, J.* e no do autor *Henriques, P.*

```
Ramalho, J.              .....................
['Ramalho95', 'Ramalho96', ..., 'LARH09', 'RH02']

Henriques, P.              .....................
['graminteractivas1990', 'Ramalho95', ..., 'RRH02', 'ORH06', 'RH02', 'cruz09']
```

Capítulo 5

Conclusão

Para concluir, ao realizar este trabalho desenvolvemos as nossas capacidades de interpretação e tratamento de informação. A utilização de todas as ferramentas necessárias para a realização deste projecto ajudou-nos a desenvolver a nossa capacidade de filtração de informação, sendo esta competência algo muito benéfico para qualquer programador.

Capítulo 6

Código do programa

Lista-se a seguir todo o código correspondente à resolução dos exercícios propostos.

```
import re
import pygraphviz
nome = r'[A-Z][a-z\.áéãõçêóíú]+'
compactName = r'[A-Z]\.'


def processa_nome(m):
    primeiro = m[1]
    ultimo = m[2].lstrip()

    return f"{ultimo}, {primeiro[0]}."


def truncaNome(m):
    primeiro = m[1]
    segundo = m[2]
    segundo = segundo[2]
    return f"{primeiro}, {segundo}."


def removeLastLetter(m):
    primeiro = m[1]
    segundo = m[2]

    return f"{primeiro}{segundo}"


def swapLast(m):
    primeiro = m[1]
    last = m[2]
    return f"{last}, {primeiro}"
```

```
def findFirst(idlinha):
    for i in idlinha:
        if i == '{':
            return '{'
        elif i == '"':
            return '"'
        else:
            return '='


def formataNome(pessoaOK):
    novo_texto = re.sub(rf'({nome})([ ]+{nome})+', processa_nome, pessoaOK)
    novo_texto = re.sub(rf'({nome})([, ]+{nome})', truncaNome, novo_texto)
    novo_texto = re.sub(rf'({nome})([, ]+{compactName})({compactName})',
    removeLastLetter, novo_texto)
    novo_texto = re.sub(rf'({nome})([, ]+{compactName})([ ]+{nome})',
    removeLastLetter, novo_texto)
    novo_texto = re.sub(rf'({nome})([, ]+{compactName})([, ]+{compactName})',
    removeLastLetter, novo_texto)
    novo_texto = re.sub(rf'({compactName})({nome})([, ]+{compactName})', swapLast,
    novo_texto)
    return novo_texto


def addbloco(bloco, info):
    padraoLinhas = '([a-zA-Z]*[\s]*=[\s]*([{])*([\sa-zA-Z0-9áçéàÉÁãñêíâ#õóúªº
    _~\?\+\!$\'\*º:&=.,\;\\\/\(\)\-]|
    [{][\sa-zA-Z0-9áçéàÉÁãñêíâ#õóúªº
    _~\?\+\!$\'\*º:&=.,\;\\\/\(\)\-]+[}])*([}])*[ ]*,)|([a-zA-Z]*[\s]*=[\s]*(["])*
    [\sa-zA-Z0-9áçéàÉÁãñêíâ#õóúªº
    _~\?\+\!$\'\*º:{}&=.,\\;\\\/\(\)\-]+(["])*[ ]*,?)'
    padraoTag = '@[a-zA-Z]+{[a-zA-Z0-9:.-]+'
    tag = re.search(padraoTag, bloco)
    match = str(re.split('{', tag.group())[0]).title()
    chave = str(re.split('{', tag.group())[1])

    if match not in info:
        info[match] = {chave: {}}
    else:
        info[match][chave] = {}

    for linha in re.finditer(padraoLinhas, bloco):
        idlinha = re.split('=', linha.group().title())[0]
        idlinha = str(idlinha).strip()
        char = findFirst(re.split('=', linha.group())[1])
        if char == '{':
            infoLinha = re.split('{', linha.group().title(), maxsplit=1)[1]
```

```
            infoLinha = re.sub('\n', ' ', str(infoLinha))
            infoLinha = re.sub('[\s]', ' ', str(infoLinha))
            info[match][chave][idlinha] = infoLinha[:-2]
        elif char == '"':
            infoLinha = re.split('"', linha.group().title(), maxsplit=1)[1]
            infoLinha = re.sub('\n', ' ', str(infoLinha))
            infoLinha = re.sub('[\s]', ' ', str(infoLinha))
            info[match][chave][idlinha] = infoLinha
        else:
            infoLinha = re.split('=', linha.group().title(), maxsplit=1)[1]
            infoLinha = re.sub('\n', ' ', str(infoLinha))
            infoLinha = re.sub('[\s]', ' ', str(infoLinha))
            info[match][chave][idlinha] = infoLinha[:-2]

    return info


def analisa(txt):
    espacogrande = '            '
    tab = '&#9&&#9'
    f = open(txt, encoding='utf-8')
    content = f.read()
    info = {}
    conta = 0
    detetaBloco = '@[a-zA-Z]+{([\sa-zA-Z0-9áçéàÉÁãñêíâ#õóúªº_~\?\+\!$\'\*º:&=.,\;\"\\
    \/\(\)\-{}])+'
    for bloco in re.finditer(detetaBloco, content):
        conta += 1
        info = addbloco(bloco.group(), info)

    for tag in info:
        for chave in info[tag]:
            for idx in info[tag][chave]:
                if idx == 'Author':
                    nospace = info[tag][chave][idx]
                    nospace = nospace.lstrip()
                    info[tag][chave][idx] = nospace
                    if info[tag][chave][idx][0] == '"' or info[tag][chave][idx][0] ==
                    '{' and not info[tag][chave][

                        nochar = info[tag][chave][idx][1:]
                        nochar = nochar.lstrip()
                        info[tag][chave][idx] = nochar
                    if info[tag][chave][idx][-1] == ',' or info[tag][chave][idx][-1]
                    == '"':
                        nochar = info[tag][chave][idx][:-1]
                        nochar = nochar.lstrip()
                        info[tag][chave][idx] = nochar
```

```
                if info[tag][chave][idx][-1] == ',' or info[tag][chave][idx][-1]
                == '"':
                    nochar = info[tag][chave][idx][:-1]
                    nochar = nochar.lstrip()
                    info[tag][chave][idx] = nochar
for tag in info:
    for chave in info[tag]:
        for idx in info[tag][chave]:
            if idx == 'Title':
                title = info[tag][chave][idx]
                title = title.lstrip()
                if title.count('{') != title.count('}'):
                    title = title[1:]
                    info[tag][chave][idx] = title
                if title[0] == '"':
                    title = title[1:]
                    info[tag][chave][idx] = title
                if info[tag][chave][idx][-1] == ',' or info[tag][chave][idx][-1]
                == '"':
                    nochar = info[tag][chave][idx][:-1]
                    nochar = nochar.lstrip()
                    info[tag][chave][idx] = nochar
                if info[tag][chave][idx][-1] == ',' or info[tag][chave][idx][-1]
                == '"':
                    nochar = info[tag][chave][idx][:-1]
                    nochar = nochar.lstrip()
                    info[tag][chave][idx] = nochar

dicAutores = {}
listaAutores = []
listaAutoresfinal = []

for tag in info:
    for chave in info[tag]:
        for idx in info[tag][chave]:
            if idx == 'Author':
                dicAutores[chave] = info[tag][chave][idx]

for chave in dicAutores:
    noAnd = dicAutores[chave].split("And")
    listaAutores.append(noAnd)
for elem in listaAutores:
    for pessoa in elem:
        pessoaOK = str(pessoa).strip()
        if (len(pessoaOK) > 1):
            novo_texto = formataNome(pessoaOK)

            listaAutoresfinal.append(novo_texto)
```

```
        listaAutoresfinal.sort()
        listaAutoresfinal = list(dict.fromkeys(listaAutoresfinal))

for chave in dicAutores:
    lis = []
    final = []
    for autores in dicAutores[chave].split('And'):

        lis.append(autores)
        for elem in lis:
            final.append(formataNome(str(elem).strip()))

        dicAutores[chave] = final

finalDoFinal = {}
for chave in dicAutores:
    for individux in dicAutores[chave]:
        if len(individux) > 3:
            if individux not in finalDoFinal:
                finalDoFinal[individux] = []
            lista = finalDoFinal[individux]
            lista.append(chave)
            lista = list(dict.fromkeys(lista))
            finalDoFinal[individux] = lista

dicOrdenado = {}
sortedaut = sorted(finalDoFinal.keys())
for i in sortedaut:
    for key, value in finalDoFinal.items():
        if key == i:
            dicOrdenado[key] = value
dicOrdenado['Frankberg, A.']=dicOrdenado.pop('Frankenberg, A.-Garcia')
relacoes = []
for autor in dicOrdenado:
    for chave in dicOrdenado[autor]:
        for outros in dicOrdenado:
            if chave in dicOrdenado[outros] and outros != autor:
                relacoes.append((autor,outros))
print(relacoes)
G = pygraphviz.AGraph()
for (o, d) in relacoes:
    G.add_edge(o, d)
G.draw('fdp2.png', format='png', prog='fdp')

f.close()
e = open('converte.html', 'w')
e.write('<html>\n\t<body>\n\t\t<p>\n')
for tag in info:
```

```
    e.write('\t\t\t' + tag + ':' + str(len(info[tag])) + '<br>\n')
    for chave in info[tag]:
        e.write('\t\t\t\t' + espacogrande + chave + '<br>\n')
        try:
            detailinfo = info[tag][chave]['Author']
            e.write('\t\t\t\t\t' + espacogrande + espacogrande + 'Author: ' +
            detailinfo + '<br>\n')
        except:
            pass
        try:
            detailinfo = info[tag][chave]['Title']
            e.write('\t\t\t\t\t' + espacogrande + espacogrande + 'Title: ' +
            detailinfo + '<br>\n')
        except:
            pass

e.write(
    '<br>' + '<br>' + '<br>' + '<br>' + espacogrande + espacogrande + espacogrande
    + espacogrande + 'INDICE:' + '<br>\n')
e.write('<br>' + espacogrande + espacogrande + 'AUTORES:' + espacogrande +
espacogrande + 'CHAVES:' + '<br>\n')
for autor in dicOrdenado:
    e.write('<br>' + espacogrande + autor + espacogrande + '.....................'
    + espacogrande + str(dicOrdenado[autor]) + '<br>\n')


e.write('\t\t</p>\n\t</body>\n</html>')
e.close()
```