Processamento de Linguagens

Trabalho Prático Nº1

Relatório de Desenvolvimento G57

Júlio Alves	Benjamim Coelho	Henrique Neto
(A89468)	(A89616)	(A89618)

8 de junho de 2021

Resumo

Neste relatório vamos abordar como ultrapassamos os desafios que encontrámos na resolução do enunciado 3, *BibTeXPro, Um processador de BibTeX*. Para este trabalho prático recorremos às expressões regulares, utilizando a biblioteca *RE* do Python.

Conteúdo

Capítulo 1

Introdução

Neste enunciado em concreto, tínhamos como objetivos:

- Calcular o número de entradas por categoria; apresentando a listagem em formato HTML por ordem alfabética.
- Criar um índice de autores,que mapeie cada autor nos respectivos registos identificados pela respectiva chave de citação (a 1ª palavra a seguir à chaveta,que é única em toda a BD)
- Transforme cada registo num documento JSON válido; a BD original deve ser então um documento JSON válido formado por uma lista de registos.
- Construa um Grafo que mostre, para um dado autor(definido na altura pelo utilizador) todos os autores que publicam normalmente como autor em causa. Recorrendo à linguagem DOT do GraphViz, gere um ficheiro com esse grafo de modo a que possa, posteriormente, usar uma das ferramentas que processam DOT para desenhar o dito grafo de associações de autores.

Capítulo 2

Arquitectura

2.1 Background

Independentemente da funcionalidade a ser executa, a aplicação desenvolvida começa por ler o ficheiro *BibTex* para memória. De seguida são aplicadas várias etapas de normalização como o processamento de símbolos e acentos (por exemplo `\'a` normaliza para `á`) e a limpeza de espaços redundantes.

Por fim o ficheiro é separado nas várias entradas que o constituem com recurso ao metodo *findall* e à expressão:

```
r'(@(\\@|[^@])+})'
```

De seguida todas as entradas são guardadas numa lista designada de *entrylist*. Todas as funcionalidades operam sobre esta lista e é a base para as funcionalidades do progama.

O programa encontra-se separado em três modulos. O módulo *tp1.py* contém a base do programa, ou seja é onde estão contidas todas as funcionalidades exigidas.

Por outro lado o modulo *util.py* é responsável pelos funções auxiliares ao modulo base, ou seja, é responsável por fazer as normalizações das palavras ou frases e é responsável por processar as estruturas de dados do modulo base no respetivos ficheiros de saída (*HTML*, *JSON*,etc.).

Por fim o modulo *BibTeXPro.py* implementa o menu do programa e funciona como o módulo principal do programa, ou seja, é o responsável por receber o input do utilizador. Para o executar, o utilizador pode utilizar a maneira habitual

```
python BibTeXPro.py
```

Se o fizer desta forma, o programa irá utilizar por defeito o ficheiro *exemplo-utf8.bib* fornecido pela equipa docente. A outra forma de o utilizar é através do uso de pipes. Tal como apresentado de seguida

```
cat exemplo | python BibTeXPro.py [opção] [autor]
```

Em opção, o utilizador deverá colocar a opção do menú que pretende executar, sendo as opções A,B,C ou D. No caso de desejar utilizar a opção D, o autor terá colocar o nome do autor pretendido (entre aspas).

2.2 Funcionalidades

2.2.1 Calcular nº de entradas por categoria

Esta funcionalidade começa por instanciar um dicionário que será responsável por mapear cada categoria no número de ocorrências da mesma. Posteriormente é aplicado a expressão

```
y = re.match(r'@(\w+)', entry)
```

a cada entrada, pois desta forma estamos a capturar todos os tipos de categoria, dado que todas elas são antecedidas pelo @ inicial. Seguidamente a palavra capturada é normalizada (com recurso ao método *built-in capitalize*) de maneira a que o nome das categorias seja insensível a maiúsculas e minúsculas.
De seguida o número de ocorrências da entrada lida é incrementado e o dicionário final é atualizado. Por fim, quando todas as entradas estiverem processadas, o dicionário é ordenado e escrito no formato de um ficheiro HTML com uma lista que indica quantas entradas da categoria foram lidas.

2.2.2 Criar um índice de autores

Semelhante à funcionalidade anterior esta funcionalidade começa por instanciar um dicionário que irá mapear cada autor à lista das chaves do registos em que está presente. De seguida é aplicada a expressão

```
re.match(r'@\w+{([^,]+)', entry)
```

que permite obter a chave do registo pelo primeiro grupo de captura. Posteriormente o conjunto de autores é capturado com a expressão

```
re.search(r'author[ \t]*=[ \t]*({+ *((\\}|[^}])*)}+|\" *((\\\"|[^"])*)\")', entry)
```

O objeto capturado terá a lista presente ou no segundo grupo de captura ou no quarto, visto que corresponde ás situações em que a lista esta contida dentro e respectivamente. Após ser averiguado qual o grupo correspondente á lista, a string capturada é dividida nos seus autores com o auxilio da expressão

```
re.split(r' +and +', authors)
```

É de notar que é obrigatório capturar os espaços antes de depois do *and* visto que caso contrário poderiam surgir separações indesejadas (com por exemplo no nome Sandra). Seguidamente cada autor é sujeito a uma normalização de espaços (que consiste em aplicar re.sub(r'[\n\r]+', r' ', author)).
Desta forma, para cada autor lido, a chave do registo capturada no inicio é adicionada à sua lista, no dicionário.
Por fim o dicionário é escrito no formato *HTML* como uma lista ordenada (indexada pelos utilizadores) de listas de chaves.

2.2.3 Transformar cada registo num documento JSON

Para esta funcionalidade, começamos por instanciar um dicionário que irá mapear cada categoria às entradas json correspondentes. Inicialmente, para cada entrada na entrylist retiramos expressões como *emph* com a expressão

```
re.sub(r'\\w+{([^}]+?)}', r'\1', entry)
```

De seguida, fazemos match à categoria em cada entrada, utilizando a expressão regular

```
re.match(r'@(\w+)', entry)
```

Caso seja feito match com sucesso, guardamos essa categoria numa variável, normalizámos para ser tudo em minúsculas e procedemos à procura da chave dessa entrada com a expressão regular

```
obj = re.search(r'\{(\w+),', entry)
```

Depois, escrevemos esta chave na variável json entry que é onde está a ser construída a entrada json correspondente. Seguidamente, procuramos por todos os campos desta entrada, guardando cada grupo de captura em variáveis distintas

```
for (tag, _, brackets, weird_brackets, _, marks, num) in re.findall(
        r'(\w+)[ \n\t]*=[ \n\t]*({((.|\n)*?)},|{([^}]*)}|\"([^"]*)\"|([0-9]*)),?', entry)
```

Assim, só temos de testar os valores de cada variável e caso sejam diferentes de null, guardamos os seus valores para serem inseridos na entrada json. No caso do resultado não ser numérico precisamos de certificarnos que a string não contém aspas nem strings multi-linha pois estas situações trariam problemas no formato json. Para tal, utilizamos as seguintes expressões que substituem estes carateres

```
target = re.sub(r'\\?\"', r'\\\"', target)
target = re.sub(r'\n', r'\\n', target)
```

Com isto escrevemos tudo na variável json entry e guardamos essa mesma entrada no dicionário, associada à chave correspondente (categoria). Por fim, quando todas as entradas forem lidas, cada lista do dicionário é escrita em formato json e associada á sua respectiva chave.

2.2.4 Constuir um grafo

Para esta funcionalidade em específico, o raciocínio utilizado foi percorrer a entrylist e, para cada campo autor, verificar se o autor dado como argumento pelo utilizador está presente. Se sim, criámos um nodo para cada um dos restantes autores presentes nesse campo e uma aresta que liga os dois autores. A expressão regular utilizada para capturar o campo de autores foi

```
re.search(r'author[ \t]*=[ \t]*({+ *((\\}|[^}])*)}+|\" *((\\\"|[^"])*)\")', entry)
```

Com r'[\t]*' garantimos que naquele sítio pode haver zero ou mais espaços. Com o grupo de captura apanhamos todos os valores dos atributos, quer estejam entre chavetas ou entre aspas. Reparamos que havia situações em que haviam $\backslash n$ no nome de alguns autores, pelo que procedemos a substituir os $\backslash n$ que possam existir por espaços através da expressão

```
name = re.sub(r'[\n\r\t ]+',r' ',name)
```

Depois disto separamos os diferentes autores através de

```
authors = re.split(r' +and +',authors)
```

e ficamos om uma lista de todos os autores pertencentes a esse campo. Finalizando, basta utilizar a biblioteca *GraphViz* e criar o nodo para cada autor e as arestas necessárias.

Capítulo 3

Conclusão

Quando iniciámos este trabalho prático, acreditávamos que seria algo simples e rápido, no entanto percebemos que cada funcionalidade tinha de abranger diversas particularidades, pois apesar de uma expressão regular ser capaz de resolver a maior parte dos casos necessários, não é capaz de apanhar campos que possuam alguma diferença em relação aos outros. Esta foi sem dúvida a grande dificuldade do trabalho prático e nós acreditamos ter superado este obstáculo com qualidade.

Este trabalho prático foi muito importante para termos noção da complexidade que é o tratamento de informação e deu-nos as ferramentas necessárias para aprender como tratar essa informação corretamente.

Apêndice A

Código do Programa

A.1 BibTeXPro.py

```
import sys
import tp1
import util

default_file = 'exemplo-utf8.bib'
input_stream = sys.stdin
is_stdout = True
output_stream = sys.stdout
entrylist = []


def inputTarget():
    global input_stream, entrylist
    # Read all file Content
    text = input_stream.read()
    entrylist = tp1.read_all_entries(text)


def menu():
    global input_stream, output_stream
    print("Escolha a funcionalidade que deseja executar:\n")
    print("a - Calcular o número de entradas por categoria")
    print("b - Criar um índice de autores")
    print("c - Transformar cada registo num documento JSON válido")
    print("d - Construir um Grafo")
    print("i - Mudar ficheiro de entrada")
    print("o - Mudar ficheiro de saída")
    print("e - Sair")
    print("\nFicheiro de entrada atual : " + input_stream.name)
    if is_stdout:
        print("O programa irá imprimir as funcionalidades na consola")
    else:
        print("Ficheiro de saída atual : " + output_stream.name)


def init_ui():
    global input_stream, default_file
    input_stream = open(default_file, "r", encoding='utf-8')


def ui():
    global input_stream, output_stream, is_stdout

    op = ' '
    while op != 'e':
        menu()
```

```
        op = input(">> ")
        if op == 'a':
            print("Funcionalidade a")
            call_A()
        elif op == 'b':
            print("Funcionalidade b")
            call_B()
        elif op == 'c':
            print("Funcionalidade c")
            call_C()
        elif op == 'd':
            print("Funcionalidade d")
            nome_autor = input("Indique o autor: ")
            call_D(nome_autor)
        elif op == 'i':
            file = input("Indique novo ficheiro de entrada: ")
            input_stream = open(file, "r", encoding='utf-8')
            inputTarget()
        elif op == 'o':
            file = input("Indique novo ficheiro de saída: ")
            if is_stdout:
                is_stdout = False
            else:
                output_stream.close()
            output_stream = open(file, "w", encoding='utf-8')
        elif op == 'e':
            pass
        else:
            print("Opção inválida!")
        input("\nClique no enter para continuar")


def call_A():
    global entrylist, output_stream
    util.write_dict_to_html(tp1.funcionalidade_a(entrylist), output_stream)
    output_stream.flush()


def call_B():
    global entrylist, output_stream
    util.write_dictlist_to_html(tp1.funcionalidade_b(entrylist), output_stream)
    output_stream.flush()


def call_C():
    global entrylist, output_stream
    util.write_dictlist_to_json(tp1.funcionalidade_c(entrylist), output_stream)
    output_stream.flush()


def call_D(nome_autor):
    global entrylist
    dot = tp1.funcionalidade_d(nome_autor, entrylist)
    dot.format = 'png'
    dot.unflatten().view()


# Main
argc = len(sys.argv)
uib = False

if argc == 1 or ('gui' in sys.argv):
    init_ui()
    uib = True

inputTarget()

if uib:
```

```
    ui()
else:
    if sys.argv[1] == 'a' or sys.argv[1] == 'A':
        call_A()
    elif sys.argv[1] == 'b' or sys.argv[1] == 'B':
        call_B()
    elif sys.argv[1] == 'c' or sys.argv[1] == 'C':
        call_C()
    elif sys.argv[1] == 'd' or sys.argv[1] == 'D':
        if argc > 2:
            call_D(sys.argv[2])
        else:
            print("Missing author")

input_stream.close()
output_stream.close()
```

A.2 tp1.py

```
import re
import util
from graphviz import Digraph


def read_all_entries(text):
    # Normalizations
    text = re.sub(r'^%.*$', r'', text)  # Remove comments
    text = util.normalize_alphas(text)  # Process \' and \~ into actual chars
    text = util.normalize_symbols(text)
    text = re.sub(r'[ \t]+', r' ', text)  # No redundant spaces

    result = []
    for (gp, _) in re.findall(r'(@(\\@|[^@])+})', text):  # It's not 100% correct
        result.append(gp)
    return result


# Calcular o número de entradas por categoria
# apresente a listagem em formato HTML por ordem alfabética
def funcionalidade_a(entrylist):
    res = dict()
    for entry in entrylist:
        y = re.match(r'@(\w+)', entry)
        if y:
            # the capitalize method desensitizes (upper/lower)cases
            entryclass = y.group(1).capitalize()
            if entryclass in res:
                res[entryclass] = res[entryclass] + 1
            else:
                res[entryclass] = 1
    return res


# Criar um índice de autores, que mapeie cada autor nos respectivos
# registos identificados pela respectiva chave de citação
def funcionalidade_b(entrylist):
    res = dict()
    for entry in entrylist:
        key = re.match(r'@\w+{([^,]+)', entry).group(1)
        authors = re.search(r'author[ \t]*=[ \t]*({+ *((\\}|[^}])*)}+|\" *((\\\"|[^"])*)\")', entry)
        if key and authors:
            if authors.group(2):
                authors = authors.group(2)
            else:
                authors = authors.group(4)
            authors = re.sub(r'[\n\r\t ]+', r' ', authors)
            for author in re.split(r' +and +', authors):
```

```
                author = util.stripSpaces(author)
                author = re.match(r'((.|\n)*) ?$', author).group(1)
                if author not in res:
                    res[author] = [key]
                else:
                    res[author].append(key)
    return res


# Transforme cada registo num documento JSON válido;
# a BD original deve ser então um documento JSON válido formado por uma lista de registos.
def funcionalidade_c(entrylist):
    output = dict()

    for entry in entrylist:
        entry = re.sub(r'\\w+{([^}]+?)}', r'\1', entry)

        obj = re.match(r'@(\w+)', entry)
        if obj:
            category = obj.group(1).lower()

            obj = re.search(r'\{(\w+),', entry)
            if obj:
                json_entry = '{\n\t"label" : "' + obj.group(1) + '",\n'

                for (tag, _, brackets, weird_brackets, _, marks, num) in re.findall(
                        r'(\w+)[ \n\t]*=[ \n\t]*({((.|\n)*?)},|{([^}]*)}|\"([^"]*)\"|([0-9]*)),?', entry):
                    if obj:
                        if brackets:
                            target = brackets
                        elif weird_brackets:
                            target = weird_brackets
                        elif marks:
                            target = marks
                        elif num:
                            target = num
                        else:
                            target = ''
                        if target.isnumeric():
                            json_entry += '\t"' + tag + '" : ' + target + ',\n'
                        else:
                            target = re.sub(r'\\?\"', '\\"', target)
                            target = re.sub(r'\n', r'\\n', target)
                            json_entry += '\t"' + tag + '" : "' + target + '",\n'
                json_entry = json_entry[:-2] + '\n}'

                if category not in output:
                    output[category] = [json_entry]
                else:
                    output[category].append(json_entry)
    return output


# Construa um Grafo que mostre, para um dado autor todos os autores que publicam
# normalmente com o autor em causa.
def funcionalidade_d(nome_autor, entrylist):
    dot = Digraph(name=nome_autor, strict=True)
    for entry in entrylist:
        author = re.search(r'author[ \t]*=[ \t]*({+ *((\\}|[^}])*)}+|\" *((\\\"|[^"])*)\")', entry)
        if author:
            if author.group(2):
                authors = author.group(2)
            else:
                authors = author.group(3)
            if authors:
                authors = re.sub(r'[\n\r\t ]+', r' ', authors)
                authors = re.split(r' +and +', authors)
                if nome_autor in authors:
```

```
                    for name in authors:
                        name = util.stripSpaces(name)
                        if (not (name == " ")) and (not (name.__str__ is None)) and not (name == nome_autor):
                            if not (name in dot.source):
                                dot.node(name.__str__())
                                dot.edge(nome_autor, name)
    return dot
```

A.3 util.py

```
import re


def acentos(matchobj):
    res = ''
    char = matchobj.group(1)
    if char == 'a':
        res = "á"
    elif char == 'A':
        res = "Á"
    elif char == 'e':
        res = "é"
    elif char == 'E':
        res = "É"
    elif char == 'i':
        res = "í"
    elif char == 'I':
        res = "Í"
    elif char == 'o':
        res = "ó"
    elif char == 'O':
        res = "Ó"
    elif char == 'u':
        res = "ú"
    elif char == 'U':
        res = "Ú"
    return res


def tils(matchobj):
    res = ''
    char = matchobj.group(1)
    if char == 'a':
        res = "ã"
    elif char == 'A':
        res = "Ã"
    elif char == 'o':
        res = "õ"
    elif char == 'O':
        res = "Õ"
    elif char == 'n':
        res = "ñ"
    elif char == 'N':
        res = "Ñ"
    return res


def normalize_alphas(target):
    target = re.sub(r'\\T', 'T', target)
    target = re.sub(r'\\\'([AaEeIiOoUu])', acentos, target)
    target = re.sub(r'\\~([AaOoNn])', tils, target)
    return target


def normalize_symbols(text):
    return re.sub(r'\\([$#&])', r'\1', text)
```

```
def stripSpaces(text):
    text = re.sub(r'^ +', r'', text)
    text = re.sub(r' +$', r'', text)
    return text


def write_dict_to_html(list_, file):
    file.write("<!DOCTYPE html>\n<html>\n\t<body>\n\t\t<ol>\n")
    for key in sorted(list_):
        file.write("\t\t\t<li>" + key + " : " + str(list_[key]) + "</li>\n")
    file.write('\t\t</ol>\n\t</body>\n</html>\n')


def write_dictlist_to_html(matrix, file):
    file.write("<!DOCTYPE html>\n<html>\n\t<body>\n\t\t<ol>\n")
    for key in sorted(matrix):
        file.write("\t\t\t<li>" + key + "</li>\n\t\t\t<ul>\n")
        for value in sorted(matrix[key]):
            file.write("\t\t\t\t<li>" + value + "</li>\n")
        file.write('\t\t\t</ul>\n')
    file.write('\t\t</ol>\n\t</body>\n</html>\n')


def add_padding(string, padding):
    return re.sub(r'\n', rf'\n{padding}', string)


def write_dictlist_to_json(matrix, file):
    file.write("{\n")
    donit = True
    for entry in matrix:
        if donit:
            donit = False
        else:
            file.write(',\n')
        donit2 = True
        file.write('\t"' + entry + "\" : [\n\t\t")
        for node in matrix[entry]:
            if donit2:
                donit2 = False
            else:
                file.write(',')
            file.write('' + add_padding(node, '\t\t'))
        file.write('\n\t]')
    file.write("\n}")
```

Apêndice B

Exemplos do ficheiro exemplo-utf8.bib

B.1 Funcionalidade A

```
<!DOCTYPE html>
<html>
   <body>
      <ol>
         <li>Article : 33</li>
         <li>Book : 1</li>
         <li>Incollection : 5</li>
         <li>Inproceedings : 112</li>
         <li>Mastersthesis : 1</li>
         <li>Misc : 1</li>
         <li>Phdthesis : 1</li>
         <li>Techreport : 11</li>
      </ol>
   </body>
</html>
```

B.2 Excerto da funcionalidade B

```
<!DOCTYPE html>
<html>
    <body>
        <ol>
          (...)
         <li>Damijan Rebernak</li>
         <ul>
            <li>RMHCV06</li>
            <li>RMHV06</li>
            <li>RMHVC06</li>
         </ul>
         <li>Daniela Cruz</li>
         <ul>
```

```
            <li>RMHVC06</li>
        </ul>
        <li>Daniela da Cruz</li>
        <ul>
            <li>BCVHU08</li>
            <li>CBHP09</li>
            <li>CFPBH07d</li>
            <li>CH07a</li>
            <li>CH07d</li>
            <li>CH07g</li>
            <li>CH07h</li>
            <li>CH09d</li>
            <li>CHP08a</li>
            <li>CHP08b</li>
            <li>CHP08i</li>
            <li>CHP09a</li>
            <li>CHV07</li>
            <li>CHV08ja</li>
            <li>CLH07c</li>
            <li>CPH07f</li>
            <li>CPH08c</li>
            <li>FCHGD09a</li>
            <li>FCHGD09b</li>
            <li>FCHV08</li>
            <li>FPCH08d</li>
            <li>FPCH08jb</li>
            <li>KOMPCCH2010</li>
            <li>MKCHCPO09</li>
            <li>OHCP09</li>
            <li>OPCH09a</li>
            <li>OPHC09a</li>
            <li>OPHCC09</li>
            <li>OPHCC2010</li>
            <li>PMCH08e</li>
            <li>PMCH08f</li>
            <li>PMCH08j</li>
            <li>RMHCV06</li>
            <li>cruz09</li>
        </ul>
        <li>Diana Santos</li>
        <ul>
            <li>linguateca</li>
        </ul>
        (...)
    </ol>
  </body>
</html>
```

B.3 Excerto da funcionalidade C

```
{
   "inproceedings" : [
      {
         "label" : "graminteractivas1990",
         "author" : "F. Mário Martins and J.J. Almeida and P.R. Henriques",
         "title" : "Mecanismos para Especifição e Prototipagem de Interfaces\n
                                                        Utilizador-Sistema",
         "note" : "(Gramáticas Interactivas guardadas)",
         "booktitle" : "3   Encontro Portugu s de Computação Gráfica",
         "address" : "Coimbra",
         "year" : 1990
      },{
         "label" : "Almeida94b",
         "author" : "J.J. Almeida",
         "title" : "{GPC} -- a Tool for higher-order grammar specification",
         "booktitle" : "Actas del X Congreso de Lenguajes Naturales e
                                              Leanguajes Formales, Sevilla",
         "year" : 1994,
         "url" : "http://natura.di.uminho.pt/~jj/pln/yalg3.ps.gz",
         "editor" : "Carlos Martin Vide",
         "keyword" : "DCG, grammar"
      },{
        (...)
        },{
            "label" : "Almeida96c",
            "author" : "J.J. Almeida and J.C. Ramalho",
            "title" : "From {BiBTeX} to {HTML} semantic nets",
            "institution" : "umdi",
            "year" : 1996,
            "number" : "DI-DAV-96:1:1",
            "docpage" : "http://www.di.uminho.pt/~jcr/bib/dbib.html",
            "keyword" : "PDavid, bibtex, librarian studies,html"
         }
      ],
   "article" : [
      {
         "label" : "jj96",
         "author" : "J.J. Almeida",
         "title" : "{NLlex} -- a tool to generate lexical analysers for
                                                        natural language",
         "year" : 1996,
         "month" : "Sep",
         "volume" : 19,
         "pages" : "81--90",
         "journal" : "Procesamiento del Lenguaje Natural",
```

```
            "publisher" : "Sociedade Española para el Procesamiento del
                                                        Lenguaje Natural",
               "keyword" : "jspell,morphology,lex,PLN,nllex",
            "url" : "http://natura.di.uminho.pt/~jj/pln/nllex2.ps.gz"
         },{
         (...)
         }
      ],
      (...)
   "misc" : [
      {
         "label" : "cruz09",
         "author" : "Daniela da Cruz and Nuno Oliveira and Pedro Rangel Henriques",
         "title" : "GraAL – A Grammar Analyzer",
         "publisher" : "Faculdade de Ci ncias da Universidade de Lisboa",
         "year" : 2009,
         "month" : "September",
         "address" : "Lisbon, Portugal",
         "note" : "Available at: url{http://inforum.org.pt/INForum2009/programa}"
      }
   ]
}
```


B.4 Exemplos da funcionalidade D

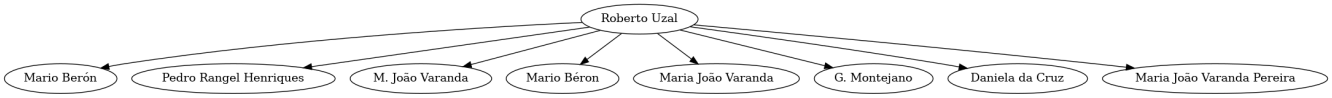

Figura B.1: Exemplo para o autor Roberto Uzal

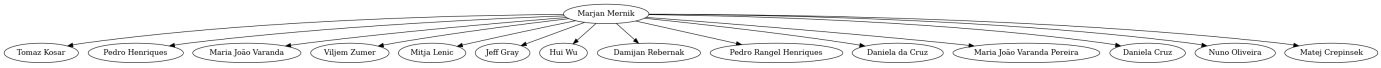

Figura B.2: Exemplo para o autor Marjan Mernik

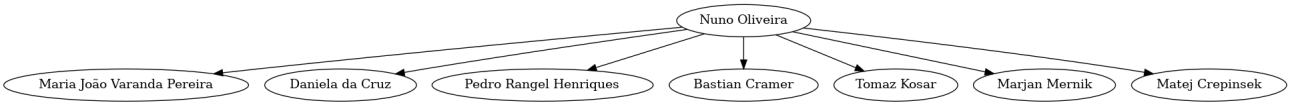

Figura B.3: Exemplo para o autor Nuno Oliveira