Processamento de Linguagens (3º ano de MIEINF)

Trabalho Prático 1

Relatório de Desenvolvimento

Luis Pereira (a77667)

Carlos Afonso (a82529)

Gonçalo Nogueira (a86617)

5 de abril de 2021

Resumo

O presente relatório descreve o trabalho prático realizado no âmbito da disciplina de *Processamento de Linguagens*, ao longo do segundo semestre do terceiro ano do Mestrado Integrado em Engenharia Informática da Universidade do Minho.

O objetivo desta fase do trabalho prático para o nosso grupo foi desenvolver o problema 3, "BibTeXPro, Um processador de BibTeX".

Este relatório inicia-se com uma breve contextualização, seguindo-se de uma descrição dos problemas propostos no enunciado e explicação da resolução desenvolvida para responder a cada um dos problemas colocados. Em cada questão apresenta-se a estrutura criada para guardar a informação essencial para uma resposta correta bem como a sua manipulação para a apresentação dos dados da forma requerida. Por fim, são apresentadas conclusões sobre o trabalho realizado.

Conteúdo

Capítulo 1

Introdução

1.1 Contextualização

O presente relatório foi elaborado para o primeiro trabalho prático da unidade curricular de Processamento de Linguagens, inserida no 2º semestre do 3º ano do Mestrado Integrado em Engenharia Informática da Universidade do Minho.

1.2 Objetivos e trabalho proposto

Pretende-se com este relatório detalhar todo o processo de tratamento de uma base de dados BibTeX. Para este trabalho foi proposto o processamento do ficheiro "exemplo-utf8.bib" que contém alguns exemplos de registos bibliográficos com o objetivo de realizar uma série de tarefas sobre o ficheiro.

1.3 Resumo do trabalho a desenvolver

Para atingir os objetivos propostos do trabalho, foi importante definir as informações que eram necessárias recolher de cada registo bibliográfico e quais as estruturas mais adequadas para o efeito. Para a recolha de informações estipulou-se as expressão regulares a serem utilizadas e as respectivas ações. Por fim, foram gerados os ficheiros de acordo com o pedido em cada tarefa.

Capítulo 2

Análise e Especificação

2.1 Plano de implementação

Após a análise do problema do enunciado e das respectivas tarefas, o grupo decidiu abordar cada tarefa individualmente, criando um programa distinto para cada tarefa de forma a ser mais direto na resolução de cada tarefa proposta e facilitação da divisão de trabalho por cada elemento.

2.2 Descrição do Problema

O problema consiste em criar um programa Python que processe uma base de dados BibTeX e que realize as seguintes tarefas:

a) Calcular o número de entradas por categoria; apresente a listagem em formato HTML por ordem alfabética.

b) 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).

c) Transformar cada registo num documento JSON válido; a BD original deve ser então um documento JSON válido formado por uma lista de registos.

d) Construa um Grafo que mostre, para um dado autor (definido na altura pelo utilizador) todos os autores que publicam normalmente com o autor em causa. Recorrendo à linguagem DOT do GraphViz7, gere um ficheiro com esse grafo de modo a que possa, posteriormente, usar uma das ferramentas que processam DOT 8 para desenhar o dito grafo de associações de autores.

Capítulo 3

Concepção/desenho da Resolução

3.1 Alínea A

Para a resolução desta tarefa a estrutura de dados utilizada foi apenas um dicionário sendo a chave o nome da categoria do registo e o valor associado um contador que indica o número de vezes que essa categoria se verificou no ficheiro fornecido.

De forma a encontrar todas as categorias distintas no ficheiro, foi pesquisado no ínicio de cada linha a existência de um @, uma vez que sabemos que o nome da categoria do registo é obrigatoriamente precedido desse caractér. Após esse valor, guarda-se num grupo todos os caractéres até ser atingido { , que denota o fim do nome da categoria e o início da chave de citação.

Sendo assim e para a concretização da metedologia em cima descrita, a seguinte expressão regular foi utilizada:

re.search(r^(.+){,linha)

Caso se verifique a expressão na linha que está a ser processada, guarda-se o grupo 1 numa variável e todos os caractéres são transformados em minísculas (com auxílio da funçao lower do python) devido á decisão tomada pelo grupo de que não iria interessar a (capitalização ?) dos nomes.

Após a conversão verifica-sa na estrutura de dados se a categoria já existe, caso exista aumenta-se o contador associado a essa categoria em uma unidade, caso contrário, essa categoria é guardada numa nova entrada no dicionário e o seu valor associado é inicializado em um.

Por fim, o método sorted do python sobre dicionários é utilizado para organizar as suas chaves, ou seja, os nomes das categorias por ordem alfabética.

Relativamente á apresentação do conteúdo em formato HTML, a forma é bastante simples, pois é simplesmente indicado no <body>um <h3>com a informação do que é pedido na tarefa e em seguida usando procede-se a uma listagem dos elementos constituintes do dicionário com a informação que obtemos previamente, onde cada entrada é mencionada usando <li>.

3.2 Alínea B

Nesta alínea, é pedido que seja criado um índice de autores, que mapeie cada autor nos respetivos registos identificados pela respetiva chave de citação. Para tal, foi seguida a seguinte estratégia. Uma vez que o ficheiro é acessado linha a linha, foram feitas restrições para averiguar o conteudo da linha. Caso a linha se refira à chave de citação, essa chave é guardada em memória. Se na linha estiver o inicio de enumeração de autores, é feita a captação até se encontrar a chaveta a finalizar essa enumeração, mesmo que esta se encontre várias linhas seguidas, de forma a ser possível captar nomes em linhas seguidas. Consoante se vai lendo a linha, se forem detetados nomes de autores, estes vão gradualmente sendo guardados em memória, mais concretamente num dicionário, que tem como índice o nome do autor, e como valores, uma lista de chaves de citação, ou seja, as chaves onde este autor está incluido. Por fim, após o ficheiro ser totalmente processado, o índice pedido é escrito para o ficheiro de texto **out.txt**

3.3 Alínea C

No desafio descrito pela alínea C foi-nos pedido que, cada publicação presente no ficheiro exemplo-utf8.bib, fosse transformada num objeto JSON válido. Para tal for aberto o ficheiro complete e aplicadas sucessivas substituiçoes recorrendo à funçào re.sub . Começando por formatar o texto original. Removendo linhas desnecessarias e new lines para posteriormente aplicar novos new lines e fixes para ir de encontro a uma formataçào JSON. Sendo finalmente apanhados os varios campos e implementados no formato correto anteriormente a remove trailing commas que possam aparecer. Posteriormente o resultado é gravado num ficheiro **out.json**

3.4 Alínea D

Para terminar, a alínea D propõem construir um grafo que associasse um dado autor a todos os seus coautores. Para esta alínea recorremos à utilização da biblioteca de Python designada Digraph, que nos permitiu escrever um ficheiro source grafo.gv e, ao mesmo tempo, escrever um outro ficheiro grafo.gv.pdf que contém a imagem do grafo correspondente, que foi produzido no formato .DOT do graphviz.

Anteriormente foi feita uma formatação em memoria, recorrendo a re.sub para reduzir a quantidade de informação com que temos de trabalhar. Deixando somente linhas de strings com os autores e coautores separados por and.

Para armazenar os dados significativos, foi criado um dicionario **authors** cujo conteudo é um set e cada chave desse set é o nome de um autor e o corpo uma lista de coautores.

A estrutura é populada por um ciclo de nested for, em que o primeiro for faz split da linha no argumento and (Tendo sido anterirmente retiradas todas as instancias que and aparece duas vezes sucessivas) criando uma lista com os nomes de autores para essa linha. Os proximos ciclos criam as chaves com o nome dos autores e populam o seu corpo com os outros autores da lista que nao sejam ele proprio. A estrutura dictionary e set asseguram que cada chave e elemento do corpo é unico, nao criando repetiçoes de elementos.

Em termos de coerência, os autores cujos nomes aparecem invertidos e separados por vírgulas foram alterados para a sua versão sem virgula. Por exemplo da Cruz, Daniela é alterado para Daniela da Cruz. Para abreviaturas de nomes, como por exemplo, J.J. Almeida. Não foram efetuadas alterações pois somos incapazes de verificar que estas abreviaturas são efetivamente os mesmo autores que algumas versões por extenso.

Em termos de interface. Cada vez que o programa é executado vai surgir um prompt para o utilizador escrever um nome de autor que seja existente na database.

Capítulo 4

Codificação e Testes

4.1 Testes realizados e Resultados

Mostram-se a seguir alguns testes feitos (valores introduzidos) e os respectivos resultados obtidos:

4.1.1 Alínea A

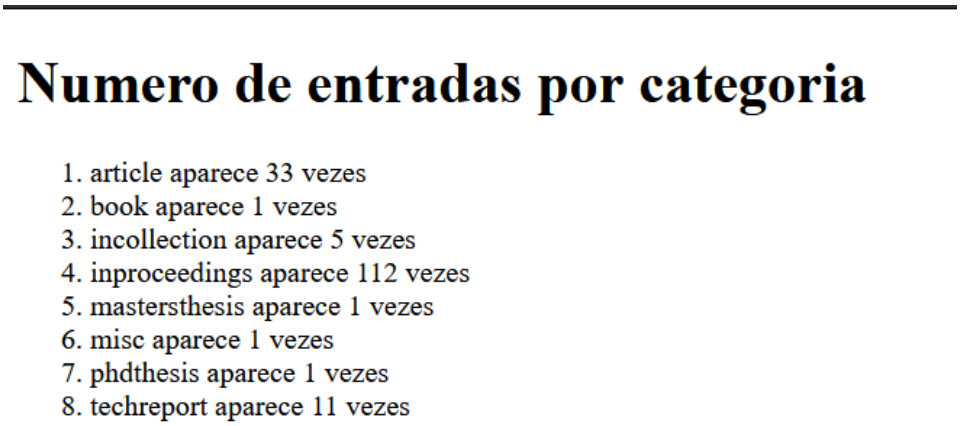

Figura 4.1: Resultado alínea A

4.1.2 Alínea B

```
F. Mário Martins->graminteractivas1990
J.J. Almeida->['graminteractivas1990', 'tlc89', 'estruturasdedados90', 'Natura', 'jspell1', 'Almeida94b',
'Ramalho95', 'Almeida96a', 'Ulisses96', 'Almeida96b', 'jj96', 'Almeida96c', 'Ramalho96', 'SGML97', 'Almeid
'RSea99', 'xmldt99', 'RRAH99', 'Barbosa2000', 'jj2001x', 'speaker:sepln2001', 'mp2001', 'alfarrabio2001',
'APL2k2.Synthesis', 'xata:xmldt', 'xata:museudapessoa', 'elpub2003', 'cp3a:terminum2003', 'sepln2003', 'sep
P.R. Henriques->['graminteractivas1990', 'Ramalho95', 'Ramalho96', 'SGML97', 'AH97', 'museums98', 'Ramalho
J.B. Barros->['tlc89', 'estruturasdedados90', 'Almeida96b', 'Barbosa2000']
{projecto Camila}->Camila
Ulisses Pinto->['jspell1', 'Almeida94c', 'Ulisses96']
L.S. Barbosa->['Barbosa95', 'Barbosa95a', 'BA97a', 'Barbosa95b', 'Barbosa2000']
J.C. Ramalho->['Ramalho95', 'Almeida96c', 'Ramalho96', 'SGML97', 'museums98', 'Ramalho98', 'RRAH99']
J.G. Rocha->['SGML97', 'museums98', 'RRAH99']
Ricardo Reis->Reis98
Barbosa, L.S.->['ABNO97a', 'ABNO97b', 'ABBN98']
Neves, F.L.->['ABNO97a', 'ABNO97b']
Oliveira, J.N.->['ABNO97a', 'ABNO97b']
M.R. Henriques->museums98
J.L.  Faria->museums98
Almeida, J.J.->ABBN98
Barros, J.B.->ABBN98
Neves, L.F.->ABBN98
Jorge Rocha->['Gis99', 'RPA99']
Ana Silva->['Gis99', 'RSea99']
Ricardo Henriques->Gis99
Pedro Henriques->['Gis99', 'KMHVZ04', 'HVMLGW05', 'HKMVZ03', 'FCHV08', 'CHP09a', 'CH09d']
Tiago Pedroso->RPA99
Jorge Gustavo Rocha->['RSea99', 'RARH98']
Mario Ricardo Henriques->RSea99
Pedro Rangel Henriques->['RSea99', 'RRH02', 'ORH06', 'RMHV06', 'RMHCV06', 'BH98', 'RAH98', 'RARH98', 'GRH0
'BHVU07d', 'BHVU07c', 'BHVU07b', 'BHVU07a', 'CHLB07a', 'BHVU06a', 'BHVU06b', 'BHVUM06', 'BHVU08', 'BHV06',
'LPRH07-TM', 'LRHGT08', 'LGFSSAH08', 'LMMVRH08', 'CHV08ja', 'FPCH08jb', 'PMCH08j', 'CHV07', 'CPH07f', 'CH0
'FPCH08d', 'PMCH08e', 'PMCH08f', 'OPCH09a', 'FCHGD09a', 'ORH09a', 'LPH09a', 'LARH09', 'OPHC09a', 'OPHCC09',
José Carlos Ramalho->['xmldt99', 'RH02', 'RAH98', 'RARH98', 'GRH06', 'JGRH04', 'JGRH03', 'GRH04', 'GRH05a',
A. M. Simões->speaker:sepln2001
J. Gustavo Rocha->['mp2001', 'alfarrabio2001']
P. Rangel Henriques->['mp2001', 'alfarrabio2001']
Sónia Moreira->mp2001
Alberto Simões->['mp2001', 'alfarrabio2001', 'cp3a:terminum2003', 'cp3a:natools2003', 'xata04:tx', 'xata04
'xata06:xmlauto', 'sepln06', 'eamt06', 'lrec06', 'elpub06-t2o', 'elpub06-blind', 'avalon:jspell', 'avalon:a
Paulo A. Rocha->freq2002
Alberto M. Simões->['freq2002', 'jspell2002', 'dag2002', 'elpub2002', 'parguess2002', 'sepln2003']
Pedro R. Henriques->dag2002
J. Alves de Castro->parguess2002
```

Figura 4.2: Resultado alínea B

4.1.3 Alínea C

Pequeno excerto do ficheiro JSON produzido pelo nosso codigo para a alínea C


```
[

{
    "categoria":"inproceedings",
    "label":"graminteractivas1990",
    "author":"F. Mário Martins and J.J. Almeida and P.R. Henriques",
    "title":"Mecanismos para Especificação e Prototipagem de Interfaces Utilizador-Sistema",
    "note":"(Gramáticas Interactivas guardadas)",
    "booktitle":"3$º$ Encontro Português de Computação Gráfica",
    "address":"Coimbra",
    "year":"1990"
},

{
    "categoria":"techreport",
    "label":"tlc89",
    "author":"J.J. Almeida and J.B. Barros",
    "title":"Teoria das Linguagens ",
    "institution":"umdi",
    "type":"Texto didáctico",
    "year":"1988",
    "keyword":"compiladores"
},

{
    "categoria":"techreport",
    "label":"estruturasdedados90",
    "author":"J.B. Barros and J.J. Almeida",
    "title":"Estruturas de Dados",
    "institution":"umdi",
    "type":"Texto didáctico",
    "year":"1990",
    "keyword":"algoritmos"
},
```


Figura 4.3: Resultado alínea C

4.1.4 Alínea D

Resultado da alínea D quando efetuada a Query com o nome de autor Nuno Oliveira.

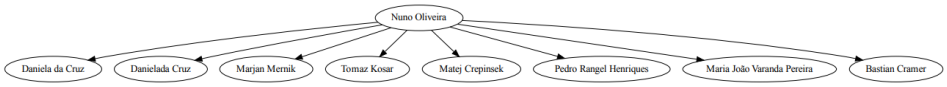

Figura 4.4: Resultado alínea D

Capítulo 5

Conclusão

Após a conclusão do trabalho proposto, o grupo avalia o resultado final como satisfatório, uma vez que todos os objetivos foram atingidos. Para além disso, adquirimos novos conhecimentos sobre expressões regulares e também da linguagem Python. Estes conhecimentos irão ser extremamente úteis no futuro, pois, por exemplo, são comuns os casos em que nos é pedido para processar grandes ficheiros, de onde apenas queremos obter a informação útil para o caso em estudo.

Apêndice A

Código do Programa

A.1 Programa A

```
import re

f = open("exemplo-utf8.bib", encoding="utf8")
print('<!DOCTYPE html>\n<html>\n<head>\n<meta charset="UTF-8">\n</head>\n<body>\n<h1>
Numero de entradas por categoria </h1>')

categorias = {}
for line in f:
    m = re.search(r'^@(.+){', line)
    if m:
        m1 = m.group(1).lower()
        if m1 in categorias:
            categorias[m1] += 1
        else:
            categorias[m1] = 1


categorias = dict(sorted(categorias.items(), key=lambda p: p[0]))

print('<ol>')
for cat in categorias:
    print(f'<li> {cat} aparece {categorias[cat]} vezes </li>')
print('</ol>\n</body>\n</html>')


f.close()
```

A.2 Programa B

```
import re



dicionarioFinal={}


f = open("exemplo-utf8.bib", encoding="utf8")
flag=True

def colocarNomes(lista , chave):

    for i in lista:
        if i in dicionarioFinal:

            # Ver se o tipo do valor é do tipo lista
            if not isinstance(dicionarioFinal[i], list):
                # Se não, converter em lista
                dicionarioFinal[i] = [dicionarioFinal[i]]
            # Append do valor à lista
            dicionarioFinal[i].append(chave)
        else:

            #Adicionar valor e chave
            if(i!=""):

                dicionarioFinal[i] = chave




for line in f:

    if(flag):
        if chave := re.search(r'@\w+{((?i:.+)),', line):
            chavez=chave.group(1)

        elif autoresCompleto := re.search(r'author *= *("|{)(.+)("|})', line):
            lista=autoresCompleto.group(2).split(" and ")
            lista = [x.strip(' ') for x in lista]
            colocarNomes(lista,chavez)

        elif autoresIncompleto := re.search(r'author *= *("|{)(.+)\n',line):
            zat=''
```

```
            zat=autoresIncompleto.group(2)
            flag=False

    else:
        if resto := re.search(r' *(.+)("|})+',line):
            zat=zat + " " +resto.group(1)
            lista=zat.split(" and" )
            lista = [x.strip(' ') for x in lista]
            colocarNomes(lista,chavez)
            zat=''
            flag=True
        else:
            resto = re.search(r' *(.+)',line)
            zat=zat + " " + resto.group(1)


t = open("out.txt", "w", encoding="utf-8")

for i in dicionarioFinal:
    t.write(str(i) + "->" + str(dicionarioFinal[i]) + "\n")
```

A.3 Programa C

```
import re

def bibtex2json(docBib):
    #abrir lista
    docBib = "[" + docBib

      # retirar linhas %
    docBib = re.sub(
        r'%(.)*',
        r'',
        docBib
    )

    # retirar linhas >
    docBib = re.sub(
        r'>.*\n+',
        r'',
        docBib
    )

    #retirar newlines
    docBib = re.sub(
        r'([À-ÿa-zA-Z,:\'\.\{\};=-])\n+',
```

```
    r'\1',
    docBib
)

#corrigir } @
docBib = re.sub(
    r'} *@',
    r'\n}\n@',
    docBib
)

#corrigir } EOF
docBib = re.sub(
    r'} *\Z',
    r'\n}\n',
    docBib
)

#corrigir #,
docBib = re.sub(
    r' *= *([0-9]+)',
    r' = "\1"',
    docBib
)

#acrescentar newlines especificos
docBib = re.sub(
    r'\",',
    r'",\n',
    docBib
)

#fix {""}
docBib = re.sub(
    r'({.*)"(.*)"(.*})',
    r'\1\2\3',
    docBib
)


docBib = re.sub(
    r'(?i)},( *(author|title|note|booktitle|year|institution|type|keyword|editor
    |url|month|abstract|pages|number|series|docpage|volume|journal|publisher
    |isbn|lang|annote|edition|shortin|isbn13|location|school|superviser|address))',
    r'},\n\1',
    docBib
```

```
)

docBib = re.sub(
    r',}',
    r',\n}',
    docBib
)

# newlines para categoria
docBib = re.sub(
    r'@',
    r'\n@',
    docBib
)

docBib = re.sub(
    r'(@[^,]+,)',
    r'\1\n',
    docBib
)


# Escape character
docBib = re.sub(
    r'\\',
    r'\\\\',
    docBib
)

# Adicionar , para manter lista
docBib = re.sub(
    r'}\n',
    r'},\n',
    docBib
)


docBib = re.sub(
    r'@([a-zA-z]+){([À-ÿa-zA-Z0-9 :,\\{}/\.\-]+),',
    r'{\n\t"categoria":"\1",\n\t"label":"\2",',
    docBib
)

docBib = re.sub(
    r'(?i) *(author|title|note|booktitle|year|institution|type|keyword|editor
    |url|month|abstract|pages|number|series|docpage|volume|journal|publisher
```

```
        |isbn|lang|annote|edition|shortin|isbn13|location|school|superviser
        |address) *=\t? *[\"{](.*)[\"}]',
        r'\t"\1":"\2"',
        docBib
    )

    #Remover double \n
    docBib = re.sub(
        r'\n\n}',
        r'\n}',
        docBib
    )

    #Remover trailing Commas
    docBib = re.sub(
        r', *\n}',
        r'\n}',
        docBib
    )

    #Arranjar espaços
    docBib = re.sub(
        r' +',
        r' ',
        docBib
    )


    docBib = docBib + "]"

    #remove last comma
    docBib = re.sub(
        r',\n]',
        r'\n]',
        docBib
    )

    return docBib


with open("exemplo-utf8.bib", encoding="utf8") as f:
    out = open ("out.json","w")
    conteudo = f.read()
    out.write(bibtex2json(conteudo))
```

A.4 Programa D

```
import re
from graphviz import Digraph

def worker(docBib,author):

     # retirar linhas %
    docBib = re.sub(
        r'%(.)*',
        r'',
        docBib
    )

    # retirar linhas >
    docBib = re.sub(
        r'>.*\n+',
        r'',
        docBib
    )

    #retirar newlines
    docBib = re.sub(
        r'([À-ÿa-zA-Z,:\'\.\{\};=-])\n+',
        r'\1',
        docBib
    )

    #corrigir } @
    docBib = re.sub(
        r'} *@',
        r'\n}\n@',
        docBib
    )

    #corrigir } EOF
    docBib = re.sub(
        r'} *\Z',
        r'\n}\n',
        docBib
    )

    #corrigir #,
    docBib = re.sub(
        r' *= *([0-9]+)',
        r' = "\1"',
```

```
    docBib
)

#acrescentar newlines especificos
docBib = re.sub(
    r'\",',
    r'",\n',
    docBib
)

#fix {""}
docBib = re.sub(
    r'({.*)"(.*)"(.*})',
    r'\1\2\3',
    docBib
)


docBib = re.sub(
    r'(?i)},( *(author|title|note|booktitle|year|institution|type|keyword|editor
    |url|month|abstract|pages|number|series|docpage|volume|journal|publisher
    |isbn|lang|annote|edition|shortin|isbn13|location|school|superviser|address))',
    r'},\n\1',
    docBib
)

docBib = re.sub(
    r',}',
    r',\n}',
    docBib
)

# newlines para categoria
docBib = re.sub(
    r'@',
    r'\n@',
    docBib
)

docBib = re.sub(
    r'(@[^,]+,)',
    r'\1\n',
    docBib
)
```

```
# Escape character
docBib = re.sub(
    r'\\',
    r'\\\\',
    docBib
)

# Remove all lines but author
docBib = re.sub(
    r'(?i)^((?!\bauthor\b).)*$\n',
    r'',
    docBib,
    flags=re.M
)

#Arranjar espaços
docBib = re.sub(
    r' +',
    r' ',
    docBib
)

#Remover double And
docBib = re.sub(
    r'and and',
    r'and',
    docBib
)


#Limpar special chars
docBib = re.sub(
    r'\\\\[\'\~]',
    r'',
    docBib
)

#Limpar special chars
docBib = re.sub(
    r'[\{\}]',
    r'"',
    docBib
)

#Limpar ,
docBib = re.sub(
```

```
        r'\",',
        r'"',
        docBib
    )

    #Remove ""
    docBib = re.sub(
        r'\"\"',
        r'"',
        docBib
    )

    #Remove projecto
    docBib = re.sub(
        r'projecto ',
        r'',
        docBib
    )

    #remove author
    docBib = re.sub(
        r'(?i) *author *= *',
        r'',
        docBib
    )

    #remove "
    docBib = re.sub(
        r'\"',
        r'',
        docBib
    )

#### Fix Names
    docBib = re.sub(
        r'( and|\n) ?(\w+), (\w+)( and |\n)',
        r'\1 \3 \2\4',
        docBib
    )

    docBib = re.sub(
        r'( and|\n) ?(\w+), (\w+)( and |\n)',
        r'\1 \3 \2\4',
        docBib
    )
```

```
docBib = re.sub(
    r' *(\w+), (\w\.\w\.)( and|\n)',
    r' \2 \1\3',
    docBib
)

docBib = re.sub(
    r'( and|\n)( ?\w+),( ?\w+)( \w+)( and |\n)',
    r'\1\3\4\2\5',
    docBib
)

docBib = re.sub(
    r'( and|\n)( \w+)( \w+),( \w+)( and|\n)',
    r'\1\4\2\3\5',
    docBib
)

docBib = re.sub(
    r'( and|\n) ?( \w+),( \w+)( \w+ã)(\w+)( \w+)( and|\n)',
    r'\1\3\4\5\6\2\7',
    docBib
)

# remove space before '
docBib = re.sub(
    r' \'',
    r'\n',
    docBib
)

# remove space after \n
docBib = re.sub(
    r'\n *',
    r'\n',
    docBib
)

# remove space before \n
docBib = re.sub(
    r' *\n',
    r'\n',
    docBib
)

authors = {}
```

```
    for line in docBib.splitlines():
        res = re.split(
            r' *\band\b *',
            line
            )
        for elem in res:
            for value in res:
                if value != elem:
                    authors.setdefault(elem, set()).add(value)
    return authors.get(author)


with open("exemplo-utf8.bib", encoding="utf8") as f:
    val = input("Enter the author: ")
    conteudo = f.read()
    lista = worker(conteudo,val)

    dot = Digraph()
    dot.node(val,val)
    for elem in lista:
        dot.node(elem,elem)
        dot.edge(val,elem)
    dot.render('grafo.gv', view=False)
```