Processamento de Linguagens (3º ano de MIEI)

Trabalho Prático 1

Enunciado 1

Relatório de Desenvolvimento

Sofia Santos
(a89615)

Carolina Vila Chã
(a89495)

5 de abril de 2021

Resumo

Neste relatório encontra-se descrita a resolução do Enunciado 1 do Trabalho Prático 1 de 2021 para a UC Processamento de Linguagens. O foco deste trabalho é a utilização de expressões regulares (*ER*) para a descricção de padrões de frases, filtragem, e transformação de texto.

Conteúdo

Capítulo 1

Introdução

O principal objetivo deste trabalho é consolidar os conhecimentos sobre ER e Python, adquiridos na UC de Processamento de Linguagens.

Capítulo 2

Análise do Problema

2.1 Descrição

Para o enunciado 1 pretende-se trabalhar com um arquivo desportivo criado por um Organizador de Provas de Orientação, realizadas em diferentes locais, e para diferentes graus de difculdade, adaptadas a diferentes classes de participantes.
São apresentadas então várias alíneas que expõem os problemas a resolver pelo grupo.

2.2 Especificação

2.2.1 Dados

Os dados a utilizar para este projeto encontram-se no ficheiro "inscritos-form.json", fornecido pelos docentes da UC.

2.2.2 Alíneas

É-nos pedido para processar o ficheiro "inscritos-form.json"de modo a:

a) imprimir o nome (convertido para maiúsculas) de todos os concorrentes que se inscrevem como 'Individuais' e são de 'Valongo'.

b) imprimir o nome completo, o email[1] e a prova em que está inscrito cada concorrente cujo nome seja 'Paulo' ou 'Ricardo', desde que usem o GMail.

c) imprimir toda a informação dos atletas da equipe "TURBULENTOS".

d) imprimir a lista dos escalões por ordem alfabética e para cada um indicar quantos atletas estão inscritos nesse escalão.

e) gerar uma página HTML com a lista das equipes inscritas em qualquer prova, indicando o seu nome e o número dos atletas que a constituem e que se inscreveram pelo menos uma vez numa prova; essa lista deve estar ordenada por ordem decrescente do número de atletas; além disso, cada equipe deve ter um link para outra página HTML com a informação que achar interessante sobre cada atleta indicando as provas em que cada um participou.

[1] No enunciado original, era pedido o "telemóvel"− no entanto este valor não existe, pelo que se escolheu o email.

Capítulo 3

Resolução

3.1 Estruturas de Dados

Usamos várias estruturas de dados auxiliares ao longo do projeto para armazenar e manipular os dados obtidos pela leitura e *parsing* do ficheiro *.json*. A estrutura principal deste projeto é uma lista de dicionários, em que cada dicionário corresponde a um atleta e armazena a sua informação com uma organização semelhante à do ficheiro *.json*.

3.2 Algoritmo e Decisões

O código do nosso projeto está contido em dois ficheiros. O principal (`main.py`) contém o código relativo ao *parsing* do ficheiro de input e à resolução das alíneas do trabalho prático, para além de uma simples interface gráfica para poder visualizar os resultados das mesmas de forma eficaz. O outro ficheiro (`similar.py`) contém uma função auxiliar desenvolvida por nós para calcular a semelhança entre duas palavras. Devido à sua relativa complexidade e versatilidade, considerámos que seria mais lógico colocá-la num ficheiro separado.

3.2.1 *main.py*

Antes de tudo, são importados os módulos necessários à resolução deste trabalho prático, utilizados pelo nosso grupo.

```
1 import re
2 import sys
3 from datetime import datetime, timedelta
4 import similar
```

O nosso grupo decidiu ler apenas uma vez o ficheiro de *input*, colocando a sua informação relevante numa estrutura de dados, que é lida posteriormente de modo a responder às alíneas do problema. Visto que as operações de leitura de ficheiros são mais lentas do que as de leitura a partir da memória, este método revela-se mais eficiente.

Primeiramente, apresentamos ao utilizador um pequeno menu, a partir do qual pode escolher qual o ficheiro que pretende utilizar.

```
1 fp = None
2 inscritos = list()
3
4 while not fp:
```

```
    opt = input("1) Usar ficheiro predefinido (inscritos-form.json).\n\
2) Especificar ficheiro a usar.\n\n\
Escolha a opção pretendida: ").strip()
    if opt == '1':
        fp = "inscritos-form.json"
    elif opt == '2':
        fp = input("Introduza o caminho para o ficheiro que pretende utilizar: ").
            strip()
```

O ciclo `while` garante-nos que apenas pode ser introduzida uma opção válida.

De seguida, o ficheiro especificado é aberto com *encoding* UTF-8 e utilizamos funções do módulo `re` para ler o conteúdo do ficheiro e armazenar a informação que pretendemos obter do mesmo. A partir da função `re.search()` somos capazes de guardar a informação relativa aos inscritos numa variável `string_inscritos`.

```
with open(fp, encoding="utf-8") as f:
    m = re.search(r'{\s*"inscritos":\[\s*((?:.|\n)*)\]\s*}',f.read())
    if not m:
        sys.exit("Erro - Ficheiro inválido")
    string_inscritos = m.group(1)
```

Usamos ainda a função `sys.exit()` para sair do programa caso o ficheiro introduzido não seja válido. Para verificar a validade do ficheiro procuramos pela chave *"inscritos"*; se estiver presente consideramos que o ficheiro é válido, e o valor correspondente a esta chave diz respeito aos dados que vamos guardar.

A partir daqui somos capazes de percorrer a string obtida e passar os valores relativos a cada inscrito para uma lista de dicionários, neste caso a lista `inscritos`, em que cada dicionário diz respeito a uma pessoa inscrita.

```
lista_inscritos = re.split(r'}\s*,\s*{',string_inscritos.strip(" \t\n{}"))
for inscrito in lista_inscritos:
    insc_dict = dict()
    for linha in inscrito.splitlines():
        linha_stripped = linha.strip()
        if linha_stripped:
            m = re.search(r'"(\w+)"\s*:\s*"(.*)"\s*,?',linha_stripped)
            insc_dict[m.group(1)] = m.group(2)

    not_duplicate = all(any(similar.similarity(inscrito[key].lower(), insc_dict[key].
        lower()) > 2 for key in inscrito) for inscrito in inscritos)

    if not_duplicate: inscritos.append(insc_dict)
```

Primeiro dividimos a string `string_inscritos` usando a função `re.split()` com um separador correspondente ao separador usado no ficheiro *.json* para separar a informação dos inscritos. Ficamos assim com uma lista (`lista_inscritos`) em que cada elemento diz respeito à informação de um inscrito, numa string.

Depois apenas percorremos essa lista e para cada elemento, linha a linha, usamos a função `re.search()` para obter os dados relativos a cada inscrito, armazenando-os em dicionários, um por inscrito, dicionários estes que são depois agregados na lista mencionada anteriormente. Antes desta agregação, é feita a filtragem de entradas repetidas.

O grupo decidiu que apenas iria excluir a entrada de um atleta se já existisse outra entrada com os mesmos valores. Poderíamos filtrar entradas repetidas através do nome ou da data de nascimento, mas poderíamos estar a excluir pessoas com o mesmo nome e nascidas no mesmo dia, para além de que a mesma pessoa pode-se inscrever duas vezes com nomes ou datas de nascimento diferentes, algo que reparámos que acontece

no ficheiro fornecido, por isso não seria um método eficaz.

O único dado que poderia ser usado para excluir repetidos seria o endereço de email, assumindo que todas as pessoas que se inscrevessem teriam que usar um endereço de email diferente, mas tal como referimos no parágrafo anterior, no ficheiro fornecido constam vários atletas com o mesmo endereço de email, e é-nos impossível saber qual deles se inscreveu primeiro. Logo, não somos capazes de usar um dado específico como filtro de repetidos, e decidimos assim que dois atletas são a mesma pessoa apenas se tiverem todos os seus dados iguais ou quase iguais.

Uma solução proposta por nós seria a inclusão de um elemento extra no perfil de cada atleta: o Número de Cartão de Cidadão (CC) ou o NIF. Uma vez que estes números são únicos, permitiriam excluir entradas repetidas. Outra opção era proibir a inscrição de mais do que um atleta com o mesmo endereço de email, e poderíamos assim usar o endereço de email para excluir repetidos.

Para verificar se dois dados são parecidos o suficiente para serem considerados iguais, desenvolvemos um simples algoritmo que calcula a semelhança de duas palavras ou expressões. Falamos deste algoritmo em detalhe na secção 3.2.2 deste relatório.

Ficamos assim com os dados do ficheiro *.json* numa estrutura de dados organizada, o que nos vai permitir percorrê-la de forma eficiente e obter os dados necessários para responder a cada alínea. Para obter esta estrutura recorremos a Expressões Regulares, tal como nos foi pedido, e a funções e operadores *standard* do Python, em situações onde usar Expressões Regulares seria muito menos eficiente.

Alínea a

>> Imprimir o nome (convertido para maiúsculas) de todos os concorrentes que se inscrevem como 'Individuais' e são de 'Valongo'.

```
1 for inscrito in inscritos:
2     if similar.similarity(inscrito["equipa"].lower(),"individual") < 2 and (n := re.
          search(r'(?i:valongo)',inscrito["morada"])):
3         print("—", inscrito["nome"].upper())
```

Graças à estrutura de dados que criámos anteriormente, responder a esta alínea é muito simples. Apenas percorremos a lista `inscritos` e para cada inscrito verificamos se a sua equipa é *"Individual"* e se a palavra *"Valongo"* aparece na sua morada.

Tal como na filtragem inicial de dados, usamos também aqui o nosso algoritmo de semelhança. Neste caso, usamo-lo para verificar se o nome da equipa de cada inscrito é próximo o suficiente de "individual"para poder ser considerado um erro ortográfico.

Alínea b

>> Imprimir o nome completo, o endereço de email e a prova em que está inscrito cada concorrente cujo nome seja 'Paulo' ou 'Ricardo', desde que usem o GMail.

```
1     print("\nNome, email e prova dos concorrentes chamados \"Paulo\" ou \"Ricardo\"
          que usam o GMail:")
2     for inscrito in inscritos:
3         if m := re.search(r'@gmail\.com$', inscrito["email"]):
4             if o := re.search(r'(?i:Paulo|Ricardo)', inscrito["nome"]):
5                 print(f'— {inscrito["nome"]}; {inscrito["email"]}; {inscrito["prova
                      "]}')
```

Tal como a alínea anterior, esta revela-se muito simples de resolver a partir da nossa estrutura de dados. Primeiro verificamos, para cada inscrito, se o seu endereço de email acaba em *"@gmail.com"*. Se tal for o caso, verificamos depois se o seu nome contém as palavras *"Paulo"* ou *"Ricardo"*.

Decidimos contabilizar atletas que tivessem estes nomes em qualquer parte do seu nome. Por exemplo, um atleta chamado "André Ricardo"será incluído nesta pesquisa. Se quiséssemos apenas contabilizar os atletas com estes nomes como primeiro nome, apenas teríamos que trocar a função `re.search()` para `re.match()`, ou colocar um `^` no início da Expressão Regular.

Alínea c

>> Imprimir toda a informação dos atletas da equipa "TURBULENTOS".

```
    turbulentos = list()
    for inscrito in inscritos:
        if m := re.fullmatch(r'(?i:turbulentos)',inscrito["equipa"]):
            turbulentos.append(inscrito)
```

Para obter os atletas da equipa "TURBULENTOS", apenas temos de, mais uma vez, percorrer a nossa estrutura de dados e filtrar os atletas cuja equipa tenha o nome "TURBULENTOS". O *matching* é feito de forma *case insensitive*, tal como nas outras alíneas, de modo a encontrar todos os resultados que possam estar escritos de forma diferente.

Na nossa análise do ficheiro encontramos uma equipa chamada "Os Turbulentos", que no contexto deste trabalho decidimos considerar como sendo uma equipa diferente, mas que poderia ser considerada a mesma equipa, com o nome escrito de forma diferente. Para tal, apenas teríamos que, de forma semelhante à alínea anterior, trocar a função `re.fullmatch()` para `re.search()`.

```
    i = 0
    while True:
        print(f'\nNome: {turbulentos[i]["nome"]};\nData de nascimento: {turbulentos[i
            ]["dataNasc"]};\nMorada: {turbulentos[i]["morada"]};\nEmail: {turbulentos[
            i]["email"]};\nProva: {turbulentos[i]["prova"]};\nEscalão: {turbulentos[i
            ]["escalao"]}')
        print(f"\nPágina {i+1} de {len(turbulentos)}\n\n[a] ver anterior; [s] ver
            seguinte; [e] sair")
        opt = input()
        if opt == 'a' and i > 0: i -= 1
        elif opt == 's' and i < len(turbulentos) - 1: i += 1
        elif opt == 'e': break
```

Devido à grande quantidade e complexidade dos resultados desta alínea, criámos um ciclo que mostra a informação de cada atleta em "páginas", como um livro, permitindo ao utilizador visualizar toda a informação de uma forma simples e eficaz.

Alínea d

>> Imprimir a lista dos escalões por ordem alfabética e para cada um indicar quantos atletas estão inscritos nesse escalão.

```
    esc_dict = dict()
    for inscrito in inscritos:
        if re.match(r'(\w+)', inscrito["escalao"]):
            x = esc_dict.get(inscrito["escalao"],0)
```

```
5               esc_dict[inscrito["escalao"]] = x+1
6       for escalao in sorted(esc_dict):
7           print("Escalão:", escalao, "; Número de atletas:", esc_dict[escalao])
```

Nesta alínea a resolução torna-se ligeiramente mais complexa que as anteriores, uma vez que temos de contar o número de atletas pertencentes a cada escalão.

Para tal, criamos uma nova estrutura - `esc_dict` - que, para cada escalão, armazena o número de atletas pertencentes a esse escalão.

Primeiro percorremos a estrutura de dados que temos usado nas outras alíneas, e para cada atleta incrementamos em uma unidade o valor no dicionário `esc_dict` correspondente ao seu escalão. Caso ainda não exista uma entrada com esse escalão, é criada uma com o valor inicial 0.

De seguida, o dicionário é ordenado alfabeticamente e são imprimidos para o terminal os valores obtidos.

Alínea e

>> Gerar uma página HTML com a lista das equipas inscritas em qualquer prova, indicando o seu nome e o número dos atletas que a constituem e que se inscreveram pelo menos uma vez numa prova; essa lista deve estar ordenada por ordem decrescente do número de atletas; além disso, cada equipe deve ter um link para outra página HTML com a informação que achar interessante sobre cada atleta indicando as provas em que cada um participou.

```
1  equipas = dict()
2  provas = dict()
3  for i, inscrito in enumerate(inscritos):
4      prova = inscrito["prova"]
5      equipa = inscrito["equipa"].upper()
6      for eqp in equipas:
7          if similar.similarity(eqp.upper(),equipa) < 2:
8              equipa = eqp.upper()
9              break
10     if not (m := re.match(r',|N/D|S/ CLUBE', equipa)):
11         equipas.setdefault(equipa, list()).append(inscrito)
12         provas.setdefault(prova, dict()).setdefault(equipa, list()).append(inscrito)
13     else:
14         provas.setdefault(prova, dict()).setdefault("INDIVIDUAL", list()).append(
               inscrito)
```

Para esta alínea, devido à sua complexidade, criámos duas estruturas de dados auxiliares. A primeira - `equipas` - armazena, para cada equipa, os atletas que dela fazem parte, e a segunda - `provas` - armazena as equipas que participam em cada prova. Por sua vez, cada equipa contém os atletas das mesmas que participaram nas respetivas provas.

Usamos mais uma vez o algoritmo de semelhança criado por nós para verificar se dois nomes de equipa se referem à mesma equipa. Para além disso, excluímos determinados nomes considerados "inválidos", como ",", "n/d"ou "s/ clube". Como num dicionário em Python as chaves devem ter um único valor imutável, passamos os nomes das equipas para maiúsculas, de modo a tornar a capitalização igual para todos os nomes e não haver duplicados.

Com estas estruturas criadas, podemos criar os ficheiros HTML.

```
1  with open("equipas.html","w", encoding="utf-8") as f:
2      f.write("""<!DOCTYPE html>\n
3  <html>
```

```
4       <head>
5           <title>Equipas</title>
6           <link rel="stylesheet" href="style.css">
7       </head>
8       <body>
9       <h1>PROVAS</h1>
10      <div class="provas">
11  """)
12
13      for prova in provas:
14          f.write(f"""     <div class="prova">
15          <h2>{prova}</h2>
16  """)
17          for equipa in sorted(provas[prova].keys(), key=lambda x : len(provas[prova][x
                ]), reverse=True):
18              f.write(f"""         <a href="./equipas/{''.join(x for x in equipa if x.
                    isalnum())}.html">
19              <span class="equipa">
20                  <h2>{equipa if equipa != "INDIVIDUAL" else "Sem equipa"}</h2>
21                  <div class="num_a"><p>{len(provas[prova][equipa])} atleta{'s' if len(
                        provas[prova][equipa]) > 1 else ''}</p></div>
22              </span>
23          </a>
24  """)
25
26          f.write("    </div>\n")
27      f.write("""     </div>
28      </body>
29  </html>
30  """)
```

Este primeiro ficheiro diz respeito às provas e, para cada prova, apresenta as equipas e o número de atletas das mesmas que nela participaram.

Tal como pedido, as equipas são ordenadas por ordem decrescente do número de atletas.

Para além disso, é possível clicar em qualquer equipa para ir para outro ficheiro HTML, que contém a informação sobre os atletas que constituem essa equipa, incluindo as provas em que cada atleta participou.

Estes ficheiros HTML das equipas são criados da seguinte forma:

```
1 today = datetime.today()
2
3 for equipa in equipas:
4     with open(f"equipas/{''.join(x for x in equipa if x.isalnum())}.html","w",
          encoding="utf-8") as ff:
5         ff.write(f"""<!DOCTYPE html>
6     <html>
7         <head>
8             <title>{equipa}</title>
9             <link rel="stylesheet" href="style.css">
10        </head>
11        <body>
12            <h1>{"Equipa: " + equipa if equipa != "INDIVIDUAL" else "Atletas sem
                  equipa"}</h1>
13            <h2>Constituição: {len(equipas[equipa])} atleta{'s' if len(equipas[equipa
                  ]) != 1 else ''}</h2>
```

```
14              <div class="atletas">
15 """)
16
17          for atleta in equipas[equipa]:
18              try:
19                  if "ou" not in atleta["dataNasc"]:
20                      birth = datetime.strptime(atleta["dataNasc"],"%d/%m/%y")
21                      if birth > datetime.today(): birth = birth.replace(year = birth.
                            year - 100)
22              except ValueError:
23                  birth = None
24              ff.write(f"""               <div class="atleta">
25                      <h3>{atleta["nome"]}</h3>
26                      <ul>
27                          <li>Idade: {str(today.year - birth.year - ((today.month,today
                                .day) < (birth.month,birth.day))) + " anos" if birth else
                                "-"}</li>
28                          <li>Escalão: {atleta["escalao"] or "-"}</li>
29                          <li>Prova: {atleta["prova"]}</li>
30                      </ul>
31                      </div>
32 """)
33          ff.write("""          </div>
34          </body>
35      </html>
36 """)
```

Usámos o módulo *datetime* do Python para calcular a idade dos atletas com base na sua data de nascimento. Todos estes ficheiros HTML contém secções (delimitadas com a *tag* `<div>`) de modo a sermos capazes de usar ficheiros CSS para tornar os *websites* gerados mais bonitos e apelativos.

3.2.2 *similar.py*

Este algoritmo, apesar de parecer relativamente complexo, é bastante simples. Recebe duas palavras ou expressões como input e tenta calcular a semelhança entre elas. Neste caso, definimos "semelhança"como a quantidade de letras que têm em comum, e o valor devolvido pela função é decrescente, por outras palavras, quanto mais próximo de zero for, mais próximas as duas palavras são. Por exemplo, duas palavras iguais terão semelhança igual a 0, enquanto que duas palavras completamente diferentes terão semelhança igual ao comprimento da menor palavra. Pode parecer contra-intuitivo, visto que duas palavras semelhantes deveriam ter um valor alto de semelhança, mas trabalhando com uma escala decrescente somos capazes de mais facilmente detetar palavras "parecidas". Dando um exemplo mais concreto, as palavras "correr"e "acorrer"têm semelhança 1, pois apenas uma letra é diferente.

Um exemplo da utilização deste algoritmo no nosso trabalho seria a palavra "indivudual", que aparece no ficheiro disponibilizado e que consideramos ser um erro ortográfico, apesar de também poder haver uma equipa com este nome. Usando o nosso algoritmo, verificamos que a "distância"entre esta palavra e a palavra "individual"é de 1, ou seja, apenas uma letra é diferente. Assim, nas alíneas *a)* e *e)* contabilizamos atletas com o valor "indivudual"no campo "equipa"como sendo atletas sem equipa.

```
1 cache = dict()
2
3 def similarity(word1 : str, word2 : str):
4     if (word1, word2) in cache:
```

```
5           return cache[(word1,word2)]
6       if word1 == word2: return 0
7       if not word1:
8           return len(word2)
9       elif not word2:
10          return len(word1)
11      check0 = similarity(word1[1:], word2[1:])
12      if check0 == 0:
13          cache[(word1,word2)] = 1
14          return 1
15      if word1[0] == word2[0]:
16          cache[(word1,word2)] = check0
17          return check0
18      check1 = similarity(word1[1:], word2)
19      if check1 < check0:
20          cache[(word1,word2)] = 1 + check1
21          return 1 + check1
22      check2 = similarity(word1, word2[1:])
23      if check2 < check0:
24          cache[(word1,word2)] = 1 + check2
25          return 1 + check2
26      cache[(word1,word2)] = 1 + check0
27      return 1 + check0
```

Este algoritmo conta com uma cache pois, como funciona de forma recursiva, acaba por se tornar bastante lento para palavras ou expressões muito grandes. A cache ajuda-nos a reduzir de forma exponencial o seu tempo de execução.

Capítulo 4

Conclusão

A realização deste projeto permitiu consolidar a matéria lecionada não só sobre ER, mas também sobre Python. Possibilitou, especificamente, a prática e consolidação de: dicionários em python, filtragem de texto utilizando *Expressões Regulares*, o desenvolvimento de *Processadores de Linguagens Regulares*, e a utilização do módulo *'re'* e as suas funções, para além de, em geral, aumentar a nossa capacidade de escrever ER para padrões de texto e trabalhar com Python.

Apêndice A

Exemplos de Utilização

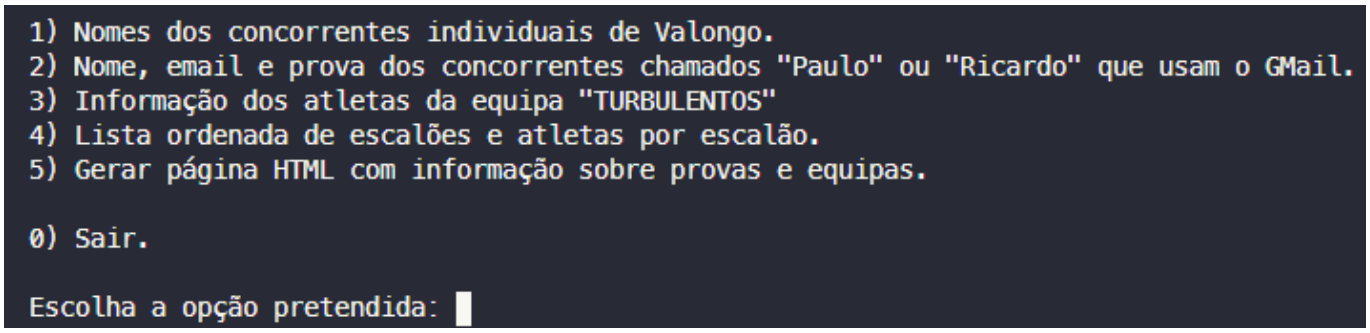

Figura A.1: Menu principal.

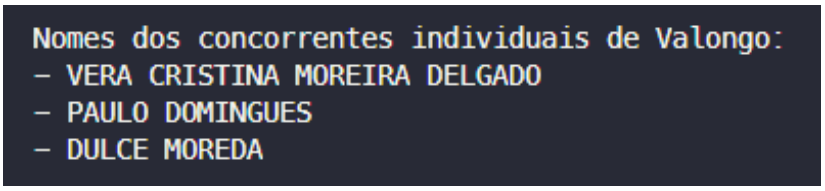

Figura A.2: Resposta à alínea *a)*.

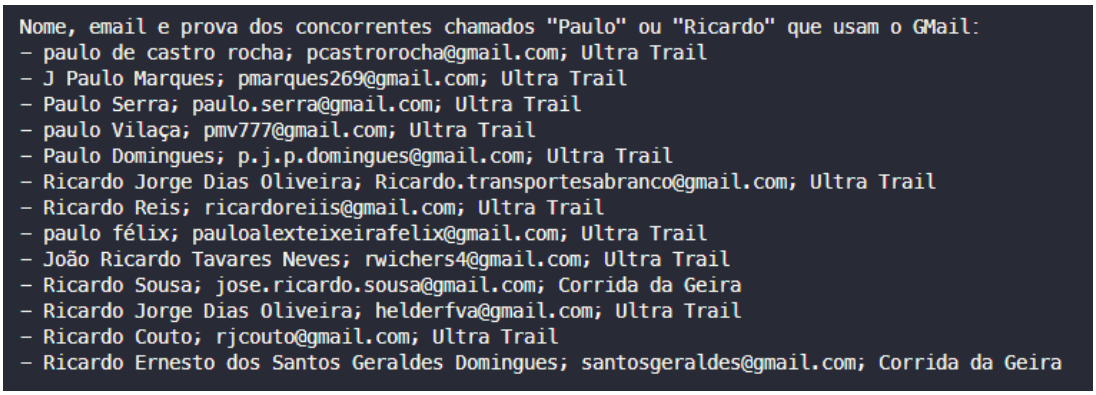

Figura A.3: Resposta à alínea *b)*.

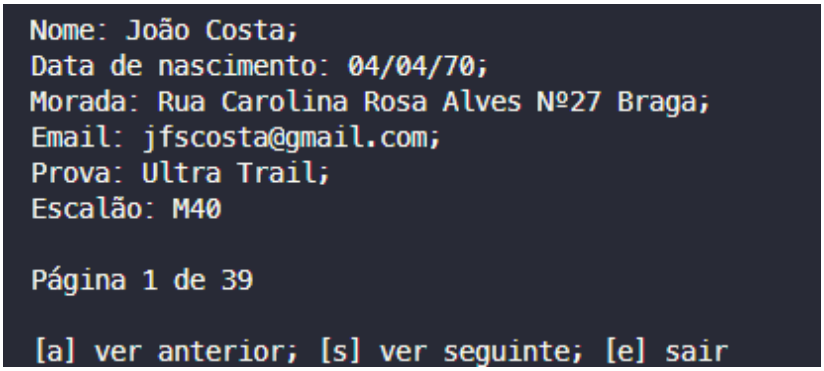

Figura A.4: Resposta à alínea *c)*.

Tal como referido anteriormente no relatório, o tamanho do resultado desta alínea levou-nos a criar um sistema de páginas, sendo possível ao utilizador ver a informação relativa a um atleta de cada vez, mas com a capacidade de percorrer os resultados de forma simples.

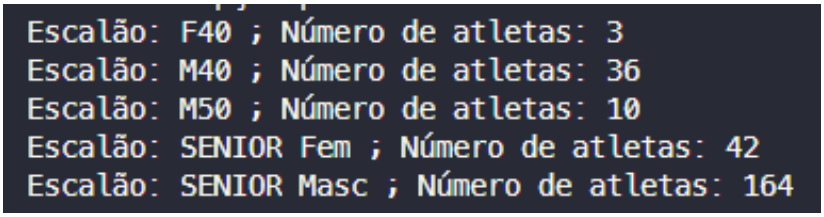

Figura A.5: Resposta à alínea *d)*.

Apêndice B

Ficheiro HTML - Equipas

Devido à grande extensão do ficheiro HTML, decidimos colocar neste relatório apenas um pequeno excerto, mas que representa a sua globalidade.

Listing B.1: Excerto do ficheiro HTML gerado

```
<!DOCTYPE html>

<html>
    <head>
        <title>Equipas</title>
        <link rel="stylesheet" href="style.css">
    </head>
    <body>
    <h1>PROVAS</h1>
    <div class="provas">
        <div class="prova">
            <h2>Ultra Trail</h2>
            <a href="./equipas/INDIVIDUAL.html">
                <span class="equipa">
                    <h2>Sem equipa</h2>
                    <div class="num_a"><p>35 atletas</p></div>
                </span>
            </a>
            <a href="./equipas/NAST.html">
                <span class="equipa">
                    <h2>NAST</h2>
                    <div class="num_a"><p>9 atletas</p></div>
                </span>
            </a>
             [...]
        </div>
    </div>
    </body>
</html>
```

Apêndice C

Ficheiro HTML - Equipa Minho Aventura

Listing C.1: Ficheiro HTML gerado

```
<!DOCTYPE html>
    <html>
        <head>
            <title>MINHO AVENTURA</title>
            <link rel="stylesheet" href="style.css">
        </head>
        <body>
            <h1>Equipa: MINHO AVENTURA</h1>
            <h2>Constituição: 2 atletas</h2>
            <div class="atletas">
            <div class="atleta">
                    <h3>Pedro Nóvoa</h3>
                    <ul>
                        <li>Idade: 58 anos</li>
                        <li>Escalão: M50</li>
                        <li>Prova: Ultra Trail</li>
                    </ul>
                    </div>
            <div class="atleta">
                    <h3>Pedro Nóvoa</h3>
                    <ul>
                        <li>Idade: 58 anos</li>
                        <li>Escalão: M50</li>
                        <li>Prova: Ultra Trail</li>
                    </ul>
                    </div>
        </div>
        </body>
    </html>
```

Apêndice D

Website gerado

Figura D.1: Página das Equipas

Figura D.2: Página dos atletas que não pertencem a nenhuma equipa

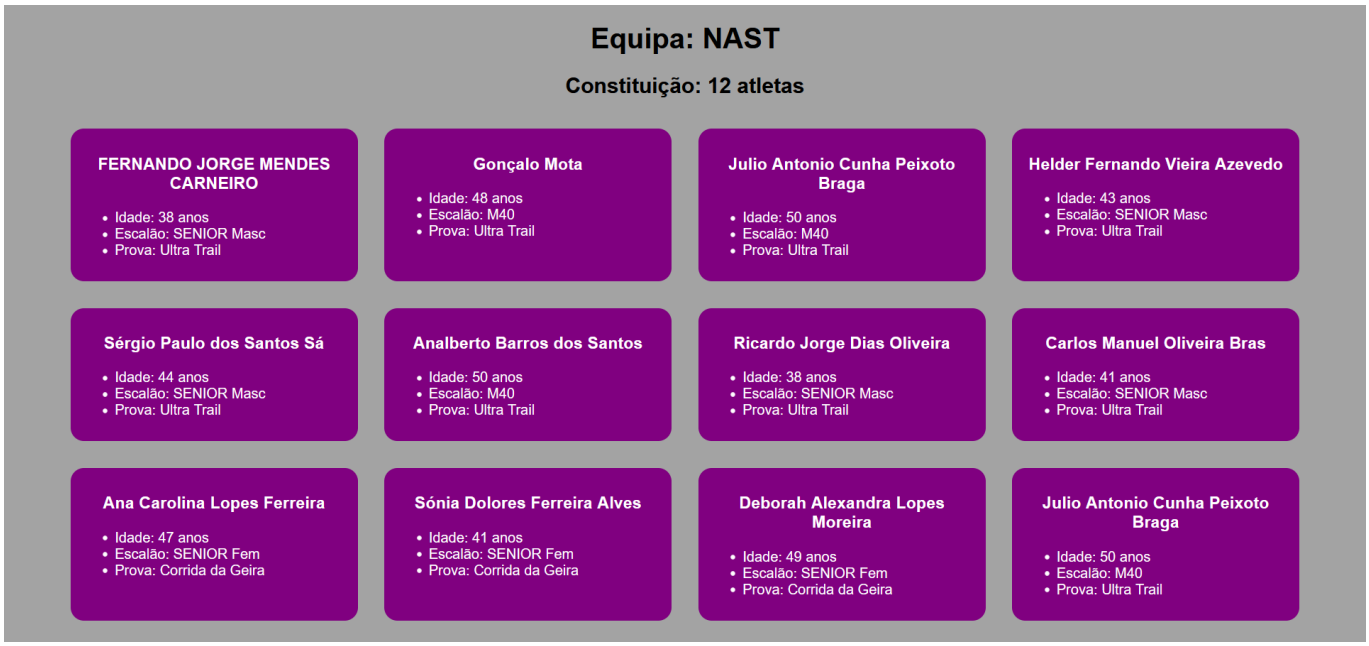

Figura D.3: Página dos atletas que pertencem à equipa NAST

Apêndice E

Código do Programa

```
import re
import sys
from datetime import datetime, timedelta
import similar

fp = None
inscritos = list()

while not fp:
    opt = input("1) Usar ficheiro predefinido (inscritos-form.json).\n\
2) Especificar ficheiro a usar.\n\n\
Escolha a opção pretendida: ").strip()
    if opt == '1':
        fp = "inscritos-form.json"
    elif opt == '2':
        fp = input("Introduza o caminho para o ficheiro que pretende utilizar: ").
            strip()

with open(fp, encoding="utf-8") as f:
    m = re.search(r'{\s*"inscritos":\[\s*((?:.|\n)*)\]\s*}',f.read())
    if not m:
        sys.exit("Erro - Ficheiro inválido")
    string_inscritos = m.group(1)

lista_inscritos = re.split(r'}\s*,\s*{',string_inscritos.strip(" \t\n{}"))
for i,inscrito in enumerate(lista_inscritos):
    print(f"Lendo ficheiro ... {round(i * 100 / len(lista_inscritos),2)}% concluído")
    insc_dict = dict()
    for linha in inscrito.splitlines():
        linha_stripped = linha.strip()
        if linha_stripped:
            m = re.search(r'"(\w+)"\s*:\s*"(.*)"\s*,?',linha_stripped)
            insc_dict[m.group(1)] = m.group(2)

    not_duplicate = all(any(similar.similarity(inscrito[key].lower(), insc_dict[key].
        lower()) > 2 for key in inscrito) for inscrito in inscritos)

    if not_duplicate: inscritos.append(insc_dict)

```

```
38 print("\nFicheiro carregado com sucesso!")
39
40 while True:
41     opt = input("""\n1) Nomes dos concorrentes individuais de Valongo.
42 2) Nome, email e prova dos concorrentes chamados \"Paulo\" ou \"Ricardo\" que usam o
       GMail.
43 3) Informação dos atletas da equipa \"TURBULENTOS\"
44 4) Lista ordenada de escalões e atletas por escalão.
45 5) Gerar página HTML com informação sobre provas e equipas.\n
46 0) Sair.\n
47 Escolha a opção pretendida: """).strip()
48     if opt == '0': break
49
50     elif opt == '1': # alinea a
51         print("\nNomes dos concorrentes individuais de Valongo:")
52         for inscrito in inscritos:
53             if similar.similarity(inscrito["equipa"].lower(),"individual") < 2 and (n
                   := re.search(r'(?i:valongo)',inscrito["morada"])):
54                 print("-", inscrito["nome"].upper())
55
56     elif opt == '2': # alinea b
57         print("\nNome, email e prova dos concorrentes chamados \"Paulo\" ou \"Ricardo
               \" que usam o GMail:")
58         for inscrito in inscritos:
59             if m := re.search(r'@gmail\.com$', inscrito["email"]):
60                 if o := re.search(r'(?i:Paulo|Ricardo)', inscrito["nome"]):
61                     print(f'- {inscrito["nome"]}; {inscrito["email"]}; {inscrito["
                           prova"]}')
62
63     elif opt == '3': # alinea c
64         turbulentos = list()
65         for inscrito in inscritos:
66             if m := re.fullmatch(r'(?i:turbulentos)',inscrito["equipa"]):
67                 turbulentos.append(inscrito)
68         i = 0
69         while True:
70             print(f'\nNome: {turbulentos[i]["nome"]};\nData de nascimento: {
                   turbulentos[i]["dataNasc"]};\nMorada: {turbulentos[i]["morada"]};\
                   nEmail: {turbulentos[i]["email"]};\nProva: {turbulentos[i]["prova"]};\
                   nEscalão: {turbulentos[i]["escalao"]}')
71             print(f"\nPágina {i+1} de {len(turbulentos)}\n\n[a] ver anterior; [s] ver
                   seguinte; [e] sair")
72             opt = input()
73             if opt == 'a' and i > 0: i -= 1
74             elif opt == 's' and i < len(turbulentos) - 1: i += 1
75             elif opt == 'e': break
76
77     elif opt == '4': # alinea d
78         esc_dict = dict()
79         for inscrito in inscritos:
80             if re.match(r'(\w+)', inscrito["escalao"]):
81                 x = esc_dict.get(inscrito["escalao"],0)
82                 esc_dict[inscrito["escalao"]] = x+1
83         for escalao in sorted(esc_dict):
```

```
84                  print("Escalão:", escalao, "; Número de atletas:", esc_dict[escalao])
85
86      elif opt == '5': # alinea e
87          today = datetime.today()
88          equipas = dict()
89          provas = dict()
90          for i, inscrito in enumerate(inscritos):
91              prova = inscrito["prova"]
92              equipa = inscrito["equipa"].upper()
93              for eqp in equipas:
94                  if similar.similarity(eqp.upper(),equipa) < 2:
95                      equipa = eqp.upper()
96                      break
97              if not (m := re.match(r',|N/D|S/ CLUBE', equipa)):
98                  equipas.setdefault(equipa, list()).append(inscrito)
99                  provas.setdefault(prova, dict()).setdefault(equipa, list()).append(
                        inscrito)
100             else:
101                 provas.setdefault(prova, dict()).setdefault("INDIVIDUAL", list()).
                        append(inscrito)
102
103 # ------------------------------------------------------------------------
104 #                       FICHEIRO PROVAS
105 # ------------------------------------------------------------------------
106
107         with open("equipas.html","w", encoding="utf-8") as f:
108             f.write("""<!DOCTYPE html>\n
109 <html>
110     <head>
111         <title>Equipas</title>
112         <link rel="stylesheet" href="style.css">
113     </head>
114     <body>
115     <h1>PROVAS</h1>
116     <div class="provas">
117 """)
118
119             for prova in provas:
120                 f.write(f"""     <div class="prova">
121         <h2>{prova}</h2>
122 """)
123                 for equipa in sorted(provas[prova].keys(), key=lambda x : len(provas[
                        prova][x]), reverse=True):
124                     f.write(f"""         <a href="./equipas/{''.join(x for x in equipa
                            if x.isalnum())}.html">
125             <span class="equipa">
126                 <h2>{equipa if equipa != "INDIVIDUAL" else "Sem equipa"}</h2>
127                 <div class="num_a"><p>{len(provas[prova][equipa])} atleta{'s' if len(
                        provas[prova][equipa]) > 1 else ''}</p></div>
128             </span>
129         </a>
130 """)
131
132                 f.write("    </div>\n")
```

```
133                 f.write("""      </div>
134         </body>
135 </html>
136 """)
137
138 # ------------------------------------------------------------------------
139 #                         FICHEIROS EQUIPAS
140 # ------------------------------------------------------------------------
141
142         for equipa in equipas:
143             with open(f"equipas/{''.join(x for x in equipa if x.isalnum())}.html","w"
                    ,encoding="utf-8") as ff:
144                 ff.write(f"""<!DOCTYPE html>
145     <html>
146         <head>
147             <title>{equipa}</title>
148             <link rel="stylesheet" href="style.css">
149         </head>
150         <body>
151             <h1>{"Equipa: " + equipa if equipa != "INDIVIDUAL" else "Atletas sem
                        equipa"}</h1>
152             <h2>Constituição: {len(equipas[equipa])} atleta{'s' if len(equipas[equipa
                        ]) != 1 else ''}</h2>
153             <div class="atletas">
154 """)
155
156                 for atleta in equipas[equipa]:
157                     try:
158                         if "ou" not in atleta["dataNasc"]:
159                             birth = datetime.strptime(atleta["dataNasc"],"%d/%m/%y")
160                             if birth > datetime.today(): birth = birth.replace(year =
                                    birth.year - 100)
161                     except ValueError:
162                         birth = None
163                     ff.write(f"""                <div class="atleta">
164                     <h3>{atleta["nome"]}</h3>
165                     <ul>
166                         <li>Idade: {str(today.year - birth.year - ((today.month,today
                                .day) < (birth.month,birth.day))) + " anos" if birth else
                                "-"}</li>
167                         <li>Escalão: {atleta["escalao"] or "-"}</li>
168                         <li>Prova: {atleta["prova"]}</li>
169                     </ul>
170                     </div>
171 """)
172                 ff.write("""          </div>
173         </body>
174     </html>
175 """)
176
177
178         print("\nFicheiro HTML gerado com sucesso!")
179
180     else:
```

```
181             continue
182     input("\nPrima ENTER para continuar.")
```