Processamento de Linguagens

Trabalho prático 1

Processador de registos de exames médicos desportivos

24 de Março, 2022

Diogo Araújo, A93313

Diogo Rebelo, Axxxxx

Joel Araújo, Axxxxx

Resumo

Serve o presente relatório como documento de apoio para o acompanhamento da resolução do enunciado relativo ao primeiro trabalho prático da cadeira de processamento de linguagens. O objetivo do mesmo é processar um ficheiro, utilizando como auxílio expressões regulares, e explorar a informação que este acarreta.

Conteúdo

1 [Introdução 4](#_Toc99035308)

2 [Descrição do problema 5](#_Toc99035309)

[2.1 Enunciado 5](#_Toc99035310)

[2.2 Características 5](#_Toc99035311)

[2.2.1 Tipo de ficheiro 5](#_Toc99035312)

[2.2.2 Indicadores estatísticos 5](#_Toc99035313)

3 [Resolução do problema 6](#_Toc99035314)

[3.1 Universo dos registos 6](#_Toc99035315)

[3.2 Organização do projeto 7](#_Toc99035316)

[3.3 Tokenizer 7](#_Toc99035317)

[3.3.1 Tokens utilizados 7](#_Toc99035318)

[3.3.2. Expressões regulares 8](#_Toc99035319)

[3.4 Estrutura de dados utilizada 11](#_Toc99035320)

[3.5 Leitura do ficheiro 12](#_Toc99035321)

Capítulo 1

# Introdução

Face à quantidade de informação que nos é disponibilizada em qualquer website e/ou documento, seja qual for o seu tipo, o olho e a capacidade de atenção humanos não fazem jus, não conseguem acompanhar e filtrar a informação que realmente necessitam de maneira fácil, muito menos eficiente. Colmatando esta improficiência, surge-nos como ferramenta expressões regulares, um mecanismo flexível e sucinto com o qual se podem identificar palavras/padrões de caracteres com interesse para o utilizador.

Com este projeto, o objetivo é melhorar a nossa aptidão para a escrita deste tipo de expressões, utilizando a linguagem Python como ponte para esse efeito.

Capítulo 2

# Descrição do problema

## 2.1 Enunciado

Dos enunciados que nos foram disponibilizados para escolha, aquele que escolhemos foi o referente a um processador de registos de exames médicos desportivos. O objetivo primário do mesmo é atuar sobre um dataset, previamente fornecido, no qual consta informação sobre indivíduos e os seus registos médicos respetivos, entre outros.

Após processar o mesmo, o trabalho foca-se numa apresentação, através de um website, de um conjunto de indicadores estatísticos, os quais apresentaremos de seguida. Destes, faz-se também acompanhar um conjunto de websites onde constam os dados dos registos que utilizamos para fazer os respetivos estudos.

## 2.2 Características

### 2.2.1 Tipo de ficheiro

Para desenvolver o nosso projeto, trabalhamos sobre um ficheiro de texto **CSV**, “emd.csv”.

### 2.2.2 Indicadores estatísticos

1. Datas extremas dos registos no dataset;
2. Distribuição por género, em cada ano e no total;
3. Distribuição por modalidade, em cada ano e no total;
4. Distribuição por idade e género (utilizando 2 escalões: < 35 anos e >= 35 anos);
5. Distribuição por morada;
6. Distribuição por estatuto de federado em cada ano;
7. Percentagem de aptos e não aptos, por ano.

Capítulo 3

# Resolução do problema

## 3.1 Universo dos registos

Como ponto de partida, o primeiro foco foi em estudar e perceber o dataset providenciado, que tipo de informação ele continha e como era apresentada. Sendo um ficheiro csv, onde os registos estão delimitados por vírgulas, deparamo-nos com os seguintes campos:

1. id
2. index
3. dataEMD
4. nome/Primeiro
5. nome/último
6. idade
7. género
8. morada
9. modalidade
10. clube
11. email
12. federado
13. resultado

Sobre este conjunto de informação, e após atentar nos estudos que nos foram indicados para apresentar, decidimos que há campos com os quais nunca nos iríamos deparar, isto é, não seriam necessários para os nossos cálculos. Esses campos são o índex, visto que ele nem é único, pois itera de 0 a 99, voltando novamente ao 0, e o clube, já que se trata de um atributo meramente decorativo para os nossos propósitos, cujo valor não é de interesse.

Ainda sobre este tipo de escolhas, debatemo-nos sobre a necessidade de trabalhar com o id e o email. O id é um atributo importante, mas em nada influenciaria nos nossos cálculos, assumimos numa primeira discussão. Contudo, se nos depararmos com registos duplicados, isso influenciaria negativamente os dados produzidos, ao que concluímos que seria de valor continuar a trabalhar com o id. Por sua vez, o email também em nada influencia, mas tendo como objetivo publicar os dados utilizados para os cálculos, entendemos por bem apresentar também o email dos indivíduos, para declarar um certo realismo e coerência na informação processada, e não parecer que são informações que poderiam ter sido inventadas por qualquer pessoa.

Deste modo, apenas o índex e o clube foram excluídos do nosso tratamento.

## 3.2 Organização do projeto

Para facilitar o modo como desenvolvemos o projeto, assim para melhor controlar onde cada parte do mesmo se encontra, dividimo-lo em três módulos, que de seguida se identificam.

* EMDsParser

Relativo a este módulo temos presente os ficheiros que tratam de definir uma classe emdRegister, que especificam o tokenizer que desenvolvemos e a respetiva leitura do ficheiro.

* Queries

Nesta porção encontram-se ficheiros destinados a responder a cada uma das alíneas que nos foram propostas para resolução, onde exploram a estrutura de dados com os dados sobre os registos obtidos.

* htmlGenerator

Por sua vez, este módulo acumula os ficheiros que estarão destinados a gerar os ficheiros html onde estarão apresentadas as nossas respostas

## 3.3 Tokenizer

Empregados de utilizar os módulos **re** e **ply** do Python, a nossa solução faz uso de um tokenizer para dividir cada linha do dataset em porções pequenas de informação com o formato pretendido, para cada campo do registo.

### 3.3.1 Tokens utilizados

Uma imagem com texto

Descrição gerada automaticamente

Munidos de nomes esclarecedores, os tokens “id”, “date”, “gender” e email ficam encarregues de capturar esses mesmos campos de cada registo. Por sua vez, o token “word” trata de recolher os campos como os nomes, moradas, modalidade e clube, visto terem formatos idênticos, enquanto o boolean acarreta com os campos de federado e resultado do exame médico, visto representarem os valores “true” ou “false”.

### 3.3.2. Expressões regulares

* t\_ID

Uma imagem com texto, dispositivo, manómetro, contador

Descrição gerada automaticamente

Após estudar o ficheiro de texto com os registos, percebemos que os ids dos registos continham sempre um padrão onde o tamanho desse id era sempre de 24 caracteres e o alfabeto utilizado apenas continha letras minúsculas e números.

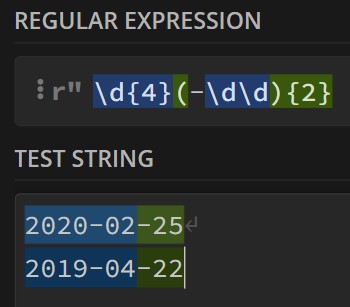
Uma imagem com texto

Descrição gerada automaticamente

* t\_DATE



O formato apresentado para o campo das datas corresponde a yyy-MM-dd.

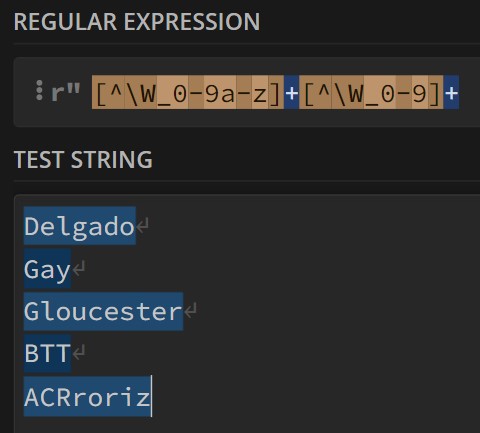


* t\_WORD



Para descrever o formato de palavras com acentos, como é o caso da língua portuguesa, poderíamos utilizar o atalho “\w” das expressões regulares, que consegue captar qualquer caracter de qualquer linguagem. Contudo, visto que as palavras que procuramos não contêm números nem “underscores”, além de terem de começar por letra maiúscula, optamos por definir o nosso padrão com aquilo que não queríamos que ele contivesse.

Por forma a captar siglas utilizadas para os nomes de algumas modalidades e/ou clubes, não limitamos o número de maiúsculas que pudessem aparecer.

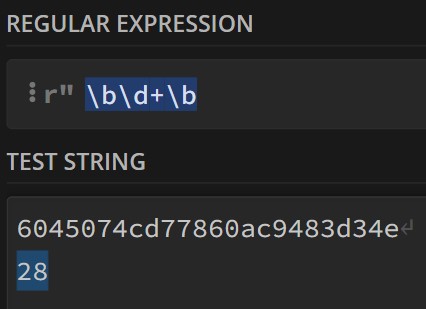


* t\_NUMBER

Uma imagem com texto, dispositivo, manómetro, contador

Descrição gerada automaticamente

Por forma a não identificar números que estivessem contidos no campo do id, por exemplo, definimos o número com “word boundary”, \b.



* t\_GENDER



O género é-nos apresentado no documento de texto com os caracteres maiúsculos ‘M’ e ‘F’. Contudo, por forma a admitir que, por lapso, o género pudesse vir identificado com as respetivas letras minúsculas, decidimos abranger também esse caso, além de especificar que essas letras tinham de se encontrar isoladas de quaisquer outros caracteres.

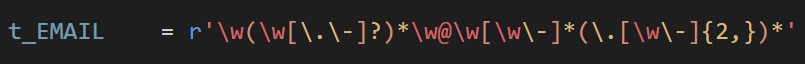
* t\_BOOLEAN

Uma imagem com texto

Descrição gerada automaticamente

Seguindo o mesmo conjunto de ideias do campo anterior, os campos de federado e de resultado médico podem ser identificados com os respetivos valores booleanos em minúsculo ou maiúsculo.

* t\_EMAIL



**Confirmar e explicar**

Por fim, atentamos que, numa primeira abordagem, consideramos as vírgulas delimitadoras dos campos de cada registo como um token a identificar. Contudo, visto que não as usaríamos, optamos por adicioná-la aos caracteres a serem ignorados pelo tokenizer, já que também não fazem parte de qualquer um dos tokens já definidos.



## 3.4 Estrutura de dados utilizada

A partir do momento que o objetivo seria explorar e produzir resultados sobre os dados que retiramos do ficheiro de texto utilizado, há necessidade de, de alguma forma, os organizar por forma a agilizar o respetivo processo.

Numa primeira abordagem, para cada registo encontrado, criamos um objeto cuja classe se identifica como emdRecord.

Uma imagem com texto

Descrição gerada automaticamente

Posteriormente, por forma a ajudar-nos nas respetivas alíneas, encontramos uma maneira de organizar estes registos que nos seria vantajosa nas posteriores pesquisas. De maneira geral, para cada ano onde haja, pelo menos, um registo, teríamos um dicionário que ligaria sempre quatro chaves, “femLT35” (*feminine lower than* 35), “femGET35” (*feminine greater or equal to* 35, “mascLT35”, “mascGET35”, cujos propósitos seriam filtrar imediatamente os registos por género e também por idade. Deste modo, para cada uma destas chaves está atribuída uma lista com todos os registos que nela se enquadram.

Exemplo:

{  
2022: { “femLT35” : [emd01, emd02], “femGET35” : [ ], “mascLT35”: [emd03], “mascGET35”: [ ] }, 2019: { “femLT35” : [emd04], “femGET35” : [ ], “mascLT35”: [emd05], “mascGET35”: [emd06 ] }   
}

## 3.5 Leitura do ficheiro

De maneira a interpretar corretamente o conteúdo que o ficheiro de texto disponibilizado continha (caracteres com acento, por exemplo), o método de abertura do mesmo especifica a flag de encoding “utf-8”.



Utilizando, por sua vez, os módulos anteriormente explicitados neste documento, o processo de leitura baseia-se em preparar a estrutura de dados criada e, percorrendo cada linha do texto, utilizar o tokenizer para preparar os chunks de informação selecionada para os fornecer a um novo objeto emdRegister. Deste modo, acedendo à data do registo escolhe-se qual é a primeira chave do dicionário a que pertence, e consultando ambas a data e a idade, encaminha-se novamente o registo para a chave do novo dicionário, adicionando-o à lista que esta contém.

Uma imagem com texto

Descrição gerada automaticamente