

UNIVERSIDADE DO MINHO

DEPARTAMENTO DE INFORMÁTICA

Machine Learning: datasets de treino  
TP1 - Exercício 5 - Grupo 79

Bruno Carvalho (A89476)  
José Gomes (A82418)  
Tiago Cunha (A87978)

5 de abril de 2021

### **Resumo**

O objetivo deste projeto é extrair informação de um dataset e posteriormente criar um site interativo com essa mesma informação.

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>2</b>
<b>2</b>	<b>Problema</b>	<b>3</b>
<b>3</b>	<b>Solução</b>	<b>4</b>
3.1	Arquitetura do Projeto . . . . .	4
<b>4</b>	<b>Manual de Utilização</b>	<b>5</b>
<b>5</b>	<b>Código</b>	<b>6</b>
5.0.1	Estrutura de Dados . . . . .	6
5.0.2	Código de Processamento . . . . .	6
5.0.3	Flask . . . . .	7
5.0.4	Templates HTML . . . . .	7
<b>6</b>	<b>Website</b>	<b>8</b>
6.0.1	Home Page . . . . .	8
6.0.2	Categoria . . . . .	9
<b>7</b>	<b>Conclusão</b>	<b>10</b>

# Capítulo 1

## Introdução

Como o projeto apontava diretamente á extração de informação de datasets, utilizamos as *expressões regulares*, na linguagem *python* de modo a agregar e extrair a informação relevante das inúmeras linhas de texto que nos foram fornecidas.

Posteriormente, fez-se uso da ferramenta *Flask* para a criação do website que utiliza a informação do dicionário criado com a informação do dataset.

## Capítulo 2

# Problema

O programa tem que cumprir os seguintes requisitos:

- Extrair vários elementos informativos de um *dataset*;
- Criar um *Website* que apresente toda a informação extraída;

O ficheiro de *input* apresentou vários problemas que precisaram de ser tratados para conseguir cumprir os requisitos propostos.

- Distinção entre linhas de informação útil e não útil;
- Captação da categoria;
- Extração do elemento da categoria e sua posição;
- Agregação de elementos cuja informação estava dispersa em várias linhas;

## Capítulo 3

# Solução

### 3.1 Arquitetura do Projeto

- Através da captação das letras *B* ou *I* no início de cada linha indicando que esta continha informação, em oposição a linhas iniciadas com a letra *O*, que contém informação irrelevante;
- Sabendo o padrão pelo qual todas as categorias são apresentadas na linha, conseguimos extrai-la facilmente;
- Como o elemento é sempre a última palavra da linha, a sua extração também é fácil e a sua posição é adquirida através de um contador;

```
v = re.search(r'^[BI]', linha)
if v:
    if v.group() == "B":
        if data != "":
            if y.group(1) in dataBase:
                dataBase[y.group(1)].append((data, str(first) + " - " + str(last)))
            else:
                dataBase[y.group(1)] = []
                dataBase[y.group(1)].append((data, str(first) + " - " + str(last)))
        data = (re.search(r'[a-zA-Z0-9]{,}$', linha)).group()
        first = counter
    else:
        data += (" " + (re.search(r'[a-zA-Z0-9]{,}$', linha)).group())
        last = counter
    y = re.search(r'-([a-zA-Z]+)', linha)
    counter += 1
```

Figura 3.1: Expressões Regulares

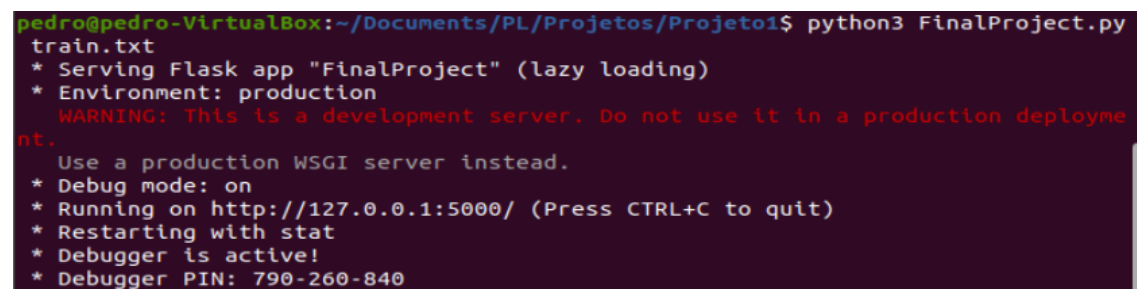
## Capítulo 4

# Manual de Utilização

Caso a ferramenta *Flask* não esteja instalada, deve proceder-se a instalação do mesmo, em ubuntu utiliza-se o seguinte código: `sudo apt install python3-flask`

Assumindo que o *Web framework Flask* está instalado o programa necessita apenas de uma da seguinte informação na linha de comandos para ser executado: `python3 FinalProject.py filename.txt`.

O *Flask* devolverá um endereço de *IP* local, através do qual conseguimos aceder ao website e visualizar os dados processados.

A terminal window with a dark background and light-colored text. The prompt is 'pedro@pedro-VirtualBox:~/Documents/PL/Projetos/Projeto1\$'. The command executed is 'python3 FinalProject.py train.txt'. The output shows Flask serving the app 'FinalProject' in production mode, with a warning about using a development server, and then starting the server on http://127.0.0.1:5000/ with debug mode on and the debugger active.

```
pedro@pedro-VirtualBox:~/Documents/PL/Projetos/Projeto1$ python3 FinalProject.py
train.txt
* Serving Flask app "FinalProject" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 790-260-840
```

Figura 4.1: Informação na linha de comandos

Uma vez na página inicial do *website* clicando numa categoria abrirá uma página com todos os elementos dessa mesma categoria. Para voltar á página inicial basta clicar novamente na categoria escolhida.

# Capítulo 5

## Código

### 5.0.1 Estrutura de Dados

O *dataBase* é um dicionário que faz a ligação entre uma categoria e uma lista de todos os seus elementos.

```
dataBase = {}  
ocurrences = []
```

Figura 5.1: Estrutura de Dados

*Ocorrences* é uma lista das categorias e o número de ocorrencias dos seus elementos.

### 5.0.2 Código de Processamento

```
def getOcurrrences():  
    for key, value in dataBase.items():  
        occurrences.append((key, len(value)))  
  
def processFile(filename):  
    f = open(filename, "r")  
    counter = 1  
    first = 0  
    last = 0  
    data = ""  
    y = ""  
    for linha in f:  
        v = re.search(r'^(BI)', linha)  
        if v:  
            if v.group() == "B":  
                if data != "":  
                    if y.group(1) in dataBase:  
                        dataBase[y.group(1)].append((data, str(first) + " - " + str(last)))  
                    else:  
                        dataBase[y.group(1)] = []  
                        dataBase[y.group(1)].append((data, str(first) + " - " + str(last)))  
                    data = (re.search(r'[a-zA-Z0-9]{,}$', linha)).group()  
                    first = counter  
                else:  
                    data += (" " + (re.search(r'[a-zA-Z0-9]{,}$', linha)).group())  
                    last = counter  
                    y = re.search(r'-([a-zA-Z]+)', linha)  
                    counter += 1  
  
            if y.group(1) in dataBase:  
                dataBase[y.group(1)].append((data, str(first) + " - " + str(last)))  
            else:  
                dataBase[y.group(1)] = []  
                dataBase[y.group(1)].append((data, str(first) + " - " + str(last)))  
    f.close()  
  
if __name__ == "__main__":  
    processFile(sys.argv[1])  
    getOcurrrences()  
    app.run(debug = True)
```

Figura 5.2: Código de processamento do ficheiro



A função `getOcorrencias` atribui o número de ocorrências dos elementos numa determinada categoria.

A `processFile` é a necessária para o resolver o problema definido.

### 5.0.3 Flask

```
app = Flask(__name__)

@app.route("/")
def home():
    return render_template("index.html", content = ocorrencias)

@app.route("/<name>")
def category(name):
    return render_template("category.html", content = DataBase.get(name.upper(), []), category = name.upper())
```

Figura 5.3: Código Flask

As funções demonstradas servem apenas para dar *render*, tanto á *Home page (home)* como a todas as categorias (*category*).

### 5.0.4 Templates HTML

```
<!doctype html>
<html>
<head>
  <title>{% block title %}{% endblock %}</title>
</head>
<body>
  {% block body %}{% endblock %}
</body>
</html>
```

Figura 5.4: Template base

A imagem mostra template base pelo qual a formatação de todas as páginas será inicializada.

```
{% extends "base.html" %}

{% block title %}
  PL Projeto
{% endblock %}

{% block body %}
  <h1><a href="http://127.0.0.1:5000/">{{category}}</a></h1>
  {% for (data, ocorrencia) in content %}
    <p>{{data}}, em {{ocorrencia}}</p>
  {% endfor %}
{% endblock %}
```

Figura 5.5: Template para categorias

Código *HTML* com excertos de *Python* que lista todos os elementos de uma determinada categoria.

# Capítulo 6

## Website

### 6.0.1 Home Page

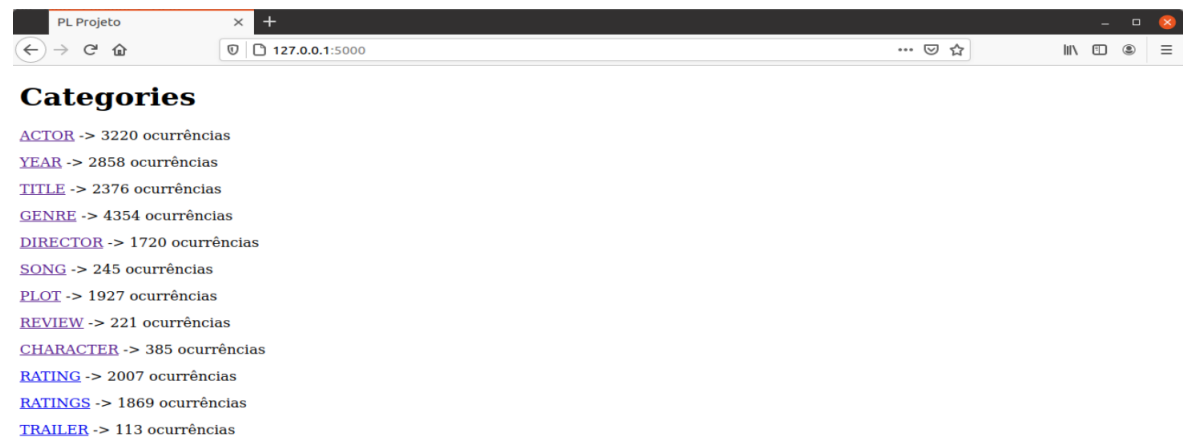


Figura 6.1: Home Page

## 6.0.2 Categoria

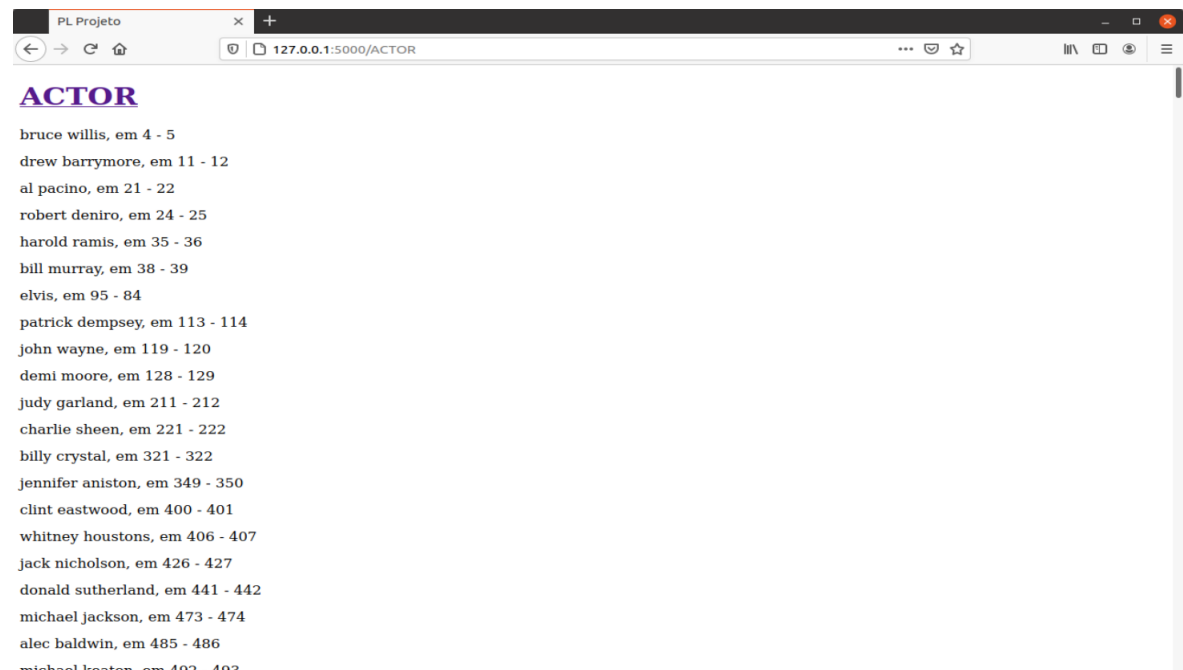


Figura 6.2: Exemplo de Categoria

## Capítulo 7

# Conclusão

Ao desenvolver este projeto conseguimos perceber a utilidade da utilização de expressões regulares para processamento de ficheiros.

Durante este projeto, exploramos a flexibilidade da linguagem *Python* e da ferramenta *Flask* para a criação de um website que, embora simples, cumpre todos os requisitos necessários.

Sucintamente, o trabalho correu bem dentro das expectativas, uma vez que todos os requisitos foram feitos tentando sempre o desenvolvimento mais eficiente possível.