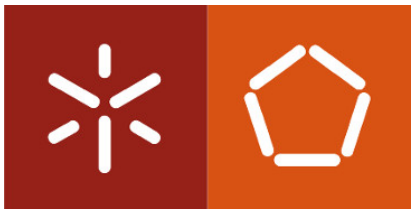

Universidade do Minho

Mestrado Integrado em Engenharia Informática

Processamento de linguagens - Trabalho Prático #1
Ano Letivo 2021/2022
Grupo 6

Gonçalo Braz (a93178) Simão Cunha (a93262)

27 de março de 2022

1 Introdução

Este relatório refere-se ao primeiro trabalho prático da UC de Processamento de Linguagens com o tema `Processadores e filtros de texto`. Foi-nos pedido que utilizássemos expressões regulares (ER) para a descrição de padrões de frases, filtragem e transformação de texto com recurso à linguagem de programação `Python`.

Assim, a equipa docente disponiblizou três enunciados, onde tinhamos que escolher um para abordarmos este tema.

Com isto, decidimos escolher o enunciado nº1: `Ficheiros CSV com listas e funções de agregação`.

2 Análise do problema

Tal como referido anteriormente, escolhemos o enunciado nº 1, onde o objetivo principal é converter um ficheiro `CSV` (*comma separated values*) para o formato `JSON`.

Como se sabe, um ficheiro `CSV` é um ficheiro de texto com um formato específico que permite que os dados sejam guardados num certo formato:

- os dados são separados por vírgulas;
- a primeira linha serve de *cabeçalho*, isto é, possui a informação a que se referem os dados nas linhas seguintes;
- cada registo irá estar dispostos linha a linha.

.

Já o ficheiro `JSON` (*JavaScript Object Notation*) é um ficheiro que contém uma série de dados estruturados em formato texto e é utilizado para transferir informações entre sistemas.

O formato aqui requerido consiste numa coleção de pares chave-valor ou chave-lista. O campo *chave* é o identificador do seu conteúdo - é representada como sendo uma *string* delimitada por aspas. Já o segundo campo pode conter um elemento do tipo *string*, tipo numérico e *boolean* ou uma lista com elementos destes tipos.

Por exemplo, considere-se o ficheiro `exemplo.csv`, cujo conteúdo é:

```
Número,Nome,Curso
3162,Cândido Faísca,Teatro
7777,Cristiano Ronaldo,Desporto
264,Marcelo Sousa,Ciência Política
```

Depois de correrm os a nossa aplicação, o resultado desejado será (com o nome `exemplo.json`):

```
[
    {
        "Número": "3612",
        "Nome": "Cândido Faísca",
        "Curso": "Teatro"
    },
    {
        "Número": "7777",
        "Nome": "Cristiano Ronaldo",
        "Curso": "Desporto"
    },
    {
        "Número": "264",
        "Nome": "Marcelo Sousa",
        "Curso": "Ciência Política"
    }
]
```

No entanto, tivemos que ter cuidado com ficheiros que têm campos que formam listas, isto é, que seguem o seguinte formato:

```
Número,Nome,Curso,Notas{5},,,,
3162,Cândido Faísca,Teatro,12,13,14,15,16
7777,Cristiano Ronaldo,Desporto,17,12,20,11,12
264,Marcelo Sousa,Ciência Política,18,19,19,20,18
```

De acordo com o enunciado, poderá aparecer listas com tamanho definido (como no exemplo acima) ou listas com um intervalo de tamanhos - é exemplo de uma linha de um ficheiro deste tipo: "`Número,Nome,Curso,Notas{3,5},,,,,`".

Além disso, tivemos que ter atenção a funções de agregação, tal como vemos no exemplo abaixo:

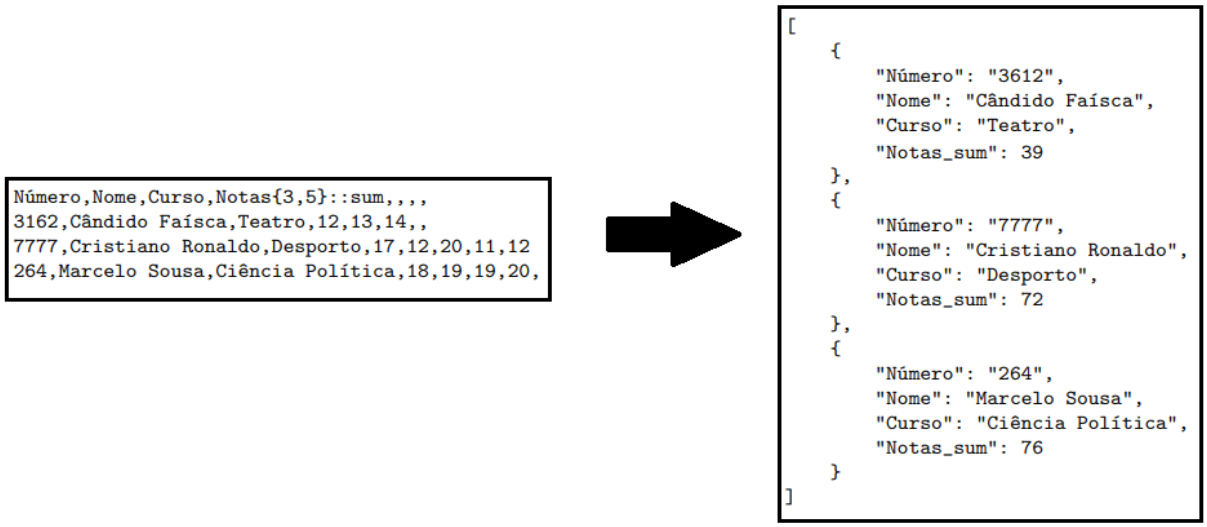

Figura 1: Transformação de `csv` para `json`

3 Resolução

De modo a criar um programa que fosse capaz de converter ficheiros do tipo `csv` em ficheiros `json` de forma eficaz e robusta seguimos o seguinte conjunto de passos:

3.1 Verificação do ficheiro

Primeiramente, teremos de fazer uma verificação e validação do ficheiro que é dado ao programa como argumento para ser convertido isto é, verificar a sua existência e validar que corresponde a um ficheiro do tipo csv, pelo que escrevemos a seguinte expressão regular que só valida ficheiros do tipo `csv`:

```
^((.+)\/)?(?P<fn>(\w+(\.+)))(?P<ext>[cC][sS][vV])$
```

Figura 2: Expressão regular para o *path* do ficheiro de *input*

3.2 Processamento do cabeçalho

Com a certeza de que estamos a trabalhar com um ficheiro `csv`, podemos começar a tratar da leitura e compreensão do cabeçalho do mesmo.

Para um cabeçalho ser válido existem algumas restrições que temos de ter em conta:

- Existem três tipos de campos, descritos pelos docentes no enunciado, que podem ser definidos: simples, lista e lista com método;
- No caso do campo simples, caso esteja vazio o `csv` não terá qualquer linha válida;
- No caso das listas, estas terão de ter à frente do seu campo um número de vírgulas com campo vazio igual ao número máximo de elementos que a lista pode ter menos 1, de modo a que o `csv` seja válido, visto que as linhas de conteúdo teriam caso contrário mais colunas do que o *header*;

Assim sendo, para reconhecer as diferentes colunas compositoras do *header*, geramos a seguinte expressão regular:

```
((?P<nome>(\w|[^",{}\n]|(\"[^"]*\"))+)((?P<n_elementos>{\d+(,\d+)?})(::(?P<metodo>(\w+)))?)?)(?P<virgulas>(,+))?(,|$)
```

Figura 3: Expressão regular para o cabeçalho

Utilizando esta expressão apanhamos coluna a coluna do *header*, tendo em conta a particularidade do uso das ”’ num `csv`, de forma a que possamos separar depois as várias colunas numa lista de colunas.

Nesta lista, as colunas que indicam a existência de listas serão guardados na forma de tuplos de dois elementos e no caso de listas com métodos serão tuplos com três elementos. Estes elementos serão, tal como se pode observar na expressão regular acima, os grupos de captura apanhados, que serão colocados no tuplo pela ordem: nome, lista e método. O grupo vírgulas é utilizado para validar a restrição acima mencionada sobre as vírgulas vazias para o caso das listas.

Chegado ao final do *header* com todas as colunas válidas, passa-se para a validação do conteúdo do `csv`, senão, o programa é terminado.

3.3 Processamento do conteúdo

Depois das validações iniciais, chegamos à análise do conteúdo do ficheiro. Neste ponto, como descrito anteriormente, possuímos o conhecimento do tipo de dados que deverá estar em cada coluna, guardado numa lista de tuplos.

Para uma linha de conteúdo ser válida existem apenas duas reservas: o número de colunas tem de ser igual ao número de elementos que tem na lista de colunas, ou seja, tem de ser igual ao número de colunas do *header* e, temos de ter em conta a mesma particularidade do uso de ”’ num ficheiro deste tipo, que desqualifica o delimitador ’,’ quando é usado dentro delas, impedindo-nos portanto de usar *.split(,)* na linha para confirmar o número de colunas.

Assim, temos de utilizar uma expressão regular para reconhecer o conteúdo das colunas. Esta vai ser similar à expressão regular do *header* que compunha o grupo `nome`:

```
((?P<column>(\w|[^",{}\n]|(\"[^"]*\"))*)(,|$))
```

Figura 4: Expressão regular para o conteúdo

A principal diferença neste caso será que esta aceita colunas vazias, visto que em alguns casos o conteúdo de um campo pode ser vazio.

Validado o número de colunas, seguimos à análise mais detalhada de cada uma, fazendo *cross-referencing* com a lista de colunas do *header*. Fazemos isto de modo a confirmar as seguintes condições:

- Campos simples, ao contrário do *header*, podem ser vazios;
- Chegado ao primeiro elemento de uma lista, temos de verificar todas as colunas pertencentes a esta seguidamente, de modo a verificar que o número de colunas preenchidas coincide com o mínimo requerido da lista, sendo que em caso negativo, a linha não será válida:

  ```
  Exemplo: Notas{3,5} (mínimo de 3 elementos)
           3,4,,5, -> 3 elementos preenchidos, lista válida
           1,2,3,4,5 -> 5 elementos preenchidos, lista válida
           3,,,, -> 1 elemento preenchidos, lista inválida
  ```

- No caso de uma lista com método, além de ter de respeitar a condição anterior, os seus elementos têm de coincidir com o tipo aceite pelo método correspondente:

  ```
  Exemplo: Notas{3,5}:sum -> todos os elementos têm de ser valores numéricos.
  ```

À medida que as colunas vão sendo validadas, de modo a não ser realizada uma nova passagem pela linha, é adicionada a uma variável da classe *line* o valor dessas colunas. Como anteriormente mencionado nas condições, as colunas referentes a uma lista são analisados de uma só vez e, do mesmo modo, a classe *line* recebe a lista inteira para guardar.

3.4 Classe *line*

A classe *line* tem como propósito guardar as colunas válidas de uma linha de forma a serem facilmente identificáveis pelo seu tipo, i.e., os elementos desta lista serão tuplos com identificadores do nome do campo, conteúdo e tipo do campo.

Colunas do tipo lista (a classe receberá todas as colunas de uma lista simultaneamente), terão todos os seus elementos guardados num único tuplo no conteúdo do mesmo, com identificador de tipo 1; colunas simples terão identificador do tipo 0 e listas de métodos terão no seu conteúdo apenas o resultado da aplicação do método na lista, com identificador do tipo 2.

Esta estruturação dos tuplos será posteriormente utilizada pelo escritor de `json` para saber quando deve aplicar certos modos de escrita, como por exemplo, entre escrever um único valor, ou escrever uma lista, onde tem de ir elemento a elemento de modo a ficar com a formatação correta.

3.5 Escrita no ficheiro `json`

O último passo que sobra na conversão será a escrita para o ficheiro `json`. Este ficheiro será criado na mesma diretoria e com o mesmo nome que o ficheiro passado como argumento ao programa, com a única diferença sendo na sua extensão.

A escrita no ficheiro será feita sempre no final do processamento de uma linha do ficheiro `csv`.

A escrita é realizada por uma única função, que receberá como argumentos o ficheiro onde irá escrever e uma variável da classe *line* que tem a informação de todas as colunas da linha.

Estas colunas embora diferentes, terão alguns tratamentos iguais. Primeiramente, todos os campos terão um nome correspondente à sua coluna no `header`. Depois, será substituído tanto do nome do campo, como do conteúdo, todas as ocorrências de ”’ internas por ”’’ de modo a ser válido no `json`.

As distinções no processamento ocorrem entre o caso dos campos simples e listas de métodos (cujo conteúdo é apenas o seu resultado), e as listas. Nos primeiros casos o conteúdo é imediatamente escrito no destino, enquanto que segundo, a lista terá de ser escrita elemento a elemento.

3.6 Métodos

Tendo em conta que poderão existir funções de agregação aplicadas a listas, decidimos que vamos permitir os seguintes cálculos com todos os elementos: somatório (**sum**), subtração (**subtr**), divisão (**div**), produtório (**prod**), média (**media**), mediana (**median**) e moda (**mode**), onde nas primeiras 4 operações implementámos o algoritmo que efetua o respetivo cálculo e nas 4 últimas recorremos ao módulo do `Python` chamado `statistics`.

Para aplicarmos uma determinada função de agregação, criamos a função da classe *lines* chamada `add_method_element` que, tendo uma lista com todos os elementos convertidos para *floats* (já temos em atenção possíveis listas com este tipo numérico de dados) e um determinado método/função (que só será aplicado caso seja uma das permitidas), aplica o resultado no ficheiro `json`. Se a lista não contiver este tipo de dados, o ficheiro resultante estará vazio. Além disso, é guardado o nome da função na forma `[nome da lista]_[nome do método]` (serão mostrados exemplos mais à frente).

3.7 *Logs*

Além do trabalho que foi descrito neste capítulo, decidimos apresentar ao utilizador alguma informação acerca do programa como trabalho extra ao pedido pela equipa docente:

- **Ficheiro de *input***: apresentamos o seu nome, o *file path* e o seu tamanho em bytes;

- **Ficheiro final**: apresentamos os dados analogamente ao ficheiro original;
- **Tempo de execução**: com este dado, possibilitamos ao utilizador saber quanto tempo demorou o seu ficheiro a ser convertido em formato `json` - consideramos relevante para saber a eficácia na conversão de ficheiros pesados;
- **Linhas validadas**: mostra o número de linhas inválidas do ficheiro `csv`, de acordo com as suas regras assentes na sua formação. Caso não haja linhas inválidas, apresentar-se-á a mensagem `Todas as linhas foram validadas`, tal como será mostrado nas figuras abaixo.

4 Exemplos de aplicação do programa

De forma a podermos fazer a demonstração do nosso conversor, decidimos criar 3 *datasets* sobre *e-sports*, sobre as estatísticas do *Mundial 2018* e sobre os alunos de uma *universidade fictícia*.

Iremos mostrar excertos dos ficheiros `csv` de teste e de ficheiros `json` formados, mas será possível consultá-los na íntegra na pasta `files` do projeto.

De forma a obter esta informação, o utilizador terá de adicionar a flag `-l`, da seguinte forma no terminal Linux, embora possa correr o programa de forma totalmente normal sem esta *flag*:

```
$python3 main.py [filepath] -l
```

4.1 Ficheiros `csv` de teste

```
"Game","ReleaseDate","Genre","TotalEarnings","OnlineEarnings","TotalPlayers","TotalTournaments"
"Age of Empires","1997","Strategy",191181.81,156839.89,261,99
"Age of Empires II","1999","Strategy",2296502.8,720999.87,1528,1140
"Age of Empires III","2005","Strategy",93913.65,41800,139,120
"Age of Empires IV","2021","Strategy",151091.75,217,248,79
"Age of Empires Online","2011","Strategy",5356.56,775,25,17
"Age of Mythology","2002","Strategy",100701,52000,129,122
"Among Us","2018","Strategy",86000,0,14,8
"Auto Chess","2019","Strategy",1144920.28,1037222.28,124,39
"Brawl Stars","2018","Strategy",2557950,1336250,215,31
"Chess.com","2007","Strategy",1975964.33,0,782,286
```

Figura 5: Dados sobre os ganhos nos *e-sports* de 1990 até 2022

```
Date,Team,Opponent,Goal Scored,Ball Possession %,Attempts,On-Target,Off-Target,Blocked,Corners,Offsides,Free Kicks,Saves,Pass Accuracy %,Passes,Distance Covered (Kms),
14-06-2018,Russia,Saudi Arabia,5,40,13,7,3,3,6,3,11,0,78,306,118,22,0,0,0,Yes,12,Group Stage,No,0,,
14-06-2018,Saudi Arabia,Russia,0,60,6,0,3,3,2,1,25,2,86,511,105,10,0,0,0,No,,Group Stage,No,0,,
15-06-2018,Egypt,Uruguay,0,43,8,3,3,2,0,1,7,3,78,395,112,12,2,0,0,No,,Group Stage,No,0,,
15-06-2018,Uruguay,Egypt,1,57,14,4,6,4,5,1,13,3,86,589,111,6,0,0,0,Yes,89,Group Stage,No,0,,
15-06-2018,Morocco,Iran,0,64,13,3,6,4,5,0,14,2,86,433,101,22,1,0,0,No,,Group Stage,No,0,1,90
15-06-2018,Iran,Morocco,1,36,8,2,5,1,2,0,22,2,86,194,100,14,3,0,0,Yes,90,Group Stage,No,0,,
15-06-2018,Portugal,Spain,3,39,8,3,2,3,4,1,13,2,87,366,102,12,1,0,0,No,4,Group Stage,No,0,,
15-06-2018,Spain,Portugal,3,61,12,5,5,2,5,3,13,0,93,727,103,10,1,0,0,Yes,24,Group Stage,No,0,,
16-06-2018,France,Australia,2,51,12,5,4,3,5,0,19,1,87,484,103,16,1,0,0,Yes,58,Group Stage,No,0,,
```

Figura 6: Dados sobre as estatísticas do *Mundial 2018* de futebol

```
Nome,Curso,Freguesia,Localidade,Notas{5}::sum,,,,
Andre Santos,Programa de Pós-Graduação em Tecnologias da Informação e Gestão em Saúde - R3,Quinta da Corriola,U.F. Carcavelos e Parede,20,12,15,10,19
Raul Pato,Programa de Residência Médica em Clínica Médica,Conceição da Abóboda,São Domingos de Rana,1,12,10,4,9
Bernardo Ronaldo,Programa de Residência Médica em Ortopedia e Traumatologia,Conceição da Abóboda,São Domingos de Rana,11,6,6,15,5
Nuno Geremias,Curso de Nutrição,Guincho,U.F. Cascais e Estoril,3,1,6,7,17
Goncalo Sousa,Curso de Fisioterapia,Murtal,São Domingos de Rana,14,14,12,3,6
Goncalo Sousa,Curso de Especialização em Técnicas Avançadas de Diagnóstico por Imagem,Serra de Sintra,Alcabideche,10,20,8,8,3
Gil Saraiva,Programa de Pós-Graduação em Tecnologias da Informação e Gestão em Saúde,Conceição da Abóboda,São Domingos de Rana,15,10,2,9,15
Amilcar Leal,Programa de Pós-Graduação em Clínica Cirúrgica,Trajouce,São Domingos de Rana,14,5,13,12,11
Patricio Jo,Programa de Residência Médica em Patologia,Areia,U.F. Cascais e Estoril,1,14,16,17,1
```

Figura 7: Dados sobre os alunos de uma *universidade fictícia*

4.2 Ficheiros `json` criados pelo programa


```
[
    {
        "\"Game\"": "\"Age of Empires\"",
        "\"ReleaseDate\"": "\"1997\"",
        "\"Genre\"": "\"Strategy\"",
        "\"TotalEarnings\"": "191181.81",
        "\"OnlineEarnings\"": "156839.89",
        "\"TotalPlayers\"": "261",
        "\"TotalTournaments\"": "99"
    },
    {
        "\"Game\"": "\"Age of Empires II\"",
        "\"ReleaseDate\"": "\"1999\"",
        "\"Genre\"": "\"Strategy\"",
        "\"TotalEarnings\"": "2296502.8",
        "\"OnlineEarnings\"": "720999.87",
        "\"TotalPlayers\"": "1528",
        "\"TotalTournaments\"": "1140"
    },
    {
        "\"Game\"": "\"Age of Empires III\"",
        "\"ReleaseDate\"": "\"2005\"",
        "\"Genre\"": "\"Strategy\"",
        "\"TotalEarnings\"": "93913.65",
        "\"OnlineEarnings\"": "41800",
        "\"TotalPlayers\"": "139",
        "\"TotalTournaments\"": "120"
    },
    {
        "\"Game\"": "\"Age of Empires IV\"",
        "\"ReleaseDate\"": "\"2021\"",
        "\"Genre\"": "\"Strategy\"",
        "\"TotalEarnings\"": "151091.75",
        "\"OnlineEarnings\"": "217",
        "\"TotalPlayers\"": "248",
        "\"TotalTournaments\"": "79"
    },
```


Figura 8: Ficheiro `json` sobre os ganhos nos *e-sports* de 1990 até 2022

Figura 9: Ficheiro `json` sobre as estatísticas do *Mundial 2018* de futebol

```
[
    {
        "Nome": "Andre Santos",
        "Curso": "Programa de Pós-Graduação em Tecnologias da Informação e Gestão em Saúde - R3",
        "Freguesia": "Quinta da Corriola",
        "Localidade": "U.F. Carcavelos e Parede",
        "Notas_sum": 76.0
    },
    {
        "Nome": "Raul Pato",
        "Curso": "Programa de Residência Médica em Clínica Médica",
        "Freguesia": "Conceição da Abóboda",
        "Localidade": "São Domingos de Rana",
        "Notas_sum": 36.0
    },
    {
        "Nome": "Bernardo Ronaldo",
        "Curso": "Programa de Residência Médica em Ortopedia e Traumatologia",
        "Freguesia": "Conceição da Abóboda",
        "Localidade": "São Domingos de Rana",
        "Notas_sum": 43.0
    },
    {
        "Nome": "Nuno Geremias",
        "Curso": "Curso de Nutrição",
        "Freguesia": "Guincho",
        "Localidade": "U.F. Cascais e Estoril",
        "Notas_sum": 34.0
    },
    {
        "Nome": "Goncalo Sousa",
        "Curso": "Curso de Fisioterapia",
        "Freguesia": "Murtal",
        "Localidade": "São Domingos de Rana",
        "Notas_sum": 49.0
    },
```


Figura 10: Ficheiro `json` sobre os alunos de uma *universidade fictícia*

4.3 *Logs* formados através dos ficheiros de teste

```
simao@DESKTOP-430FJ5B:/mnt/c/Users/utilizador/Desktop/pl_projeto/src$ python3 main.py files/fifa.csv -l
Conversor de ficheiros csv para ficheiros json - Grupo 6 | Processamento de linguagens (2021/2022)
Todas as linhas são válidas!
Ficheiro a converter: fifa.csv (12555 bytes) - encontrado na diretoria files/fifa.csv
Novo ficheiro: fifa.json (77042 bytes) - disponível em files/fifa.json
Tempo de execução: 0.05139 ms
simao@DESKTOP-430FJ5B:/mnt/c/Users/utilizador/Desktop/pl_projeto/src$ python3 main.py files/esports.csv -l
Conversor de ficheiros csv para ficheiros json - Grupo 6 | Processamento de linguagens (2021/2022)
Todas as linhas são válidas!
Ficheiro a converter: esports.csv (35751 bytes) - encontrado na diretoria files/esports.csv
Novo ficheiro: esports.json (134423 bytes) - disponível em files/esports.json
Tempo de execução: 0.03732 ms
simao@DESKTOP-430FJ5B:/mnt/c/Users/utilizador/Desktop/pl_projeto/src$ python3 main.py files/turma.csv -l
Conversor de ficheiros csv para ficheiros json - Grupo 6 | Processamento de linguagens (2021/2022)
Foram encontradas 7 linhas inválidas em 101 linhas lidas.
Ficheiro a converter: turma.csv (10953 bytes) - encontrado na diretoria files/turma.csv
Novo ficheiro: turma.json (17471 bytes) - disponível em files/turma.json
Tempo de execução: 0.01642 ms
simao@DESKTOP-430FJ5B:/mnt/c/Users/utilizador/Desktop/pl_projeto/src$
```

Figura 11: *Logs* criados

5 Conclusão

Em suma, criámos uma aplicação capaz de responder ao objetivo de converter ficheiros `csv`, do tipo simples ou com propriedades extra como as dadas no enunciado, em `json`, de forma fiável e eficiente, através do uso, como pretendido, da linguagem `Python` e do respetivo módulo para manipulação de expressões regulares, `re`, e ainda com os *logs* que informam o seu utilizador acerca do funcionamento do programa.