

Processamento de Linguagens Fase 1 CSV

André Gonçalves Pinto a93173 and Rui Pedro Chaves Lousada a93252

Universidade do Minho **Grupo 58**

Abstract. Esta fase tem como objetivo desenvolver um projeto da cadeira de **Processamento de Linguagens** capaz de identificar padrões através de ERs. Seremos capazes de descobrir qual a forma mais eficiente de guardar estruturas de dados de uma forma escalável com a complexidade do problema e de desenvolver um filtro de texto para capturar padrões e efetuar substituições de acordo.

Keywords: Regular Expressions · Python · Text Filter · State Machines
· Language Processing

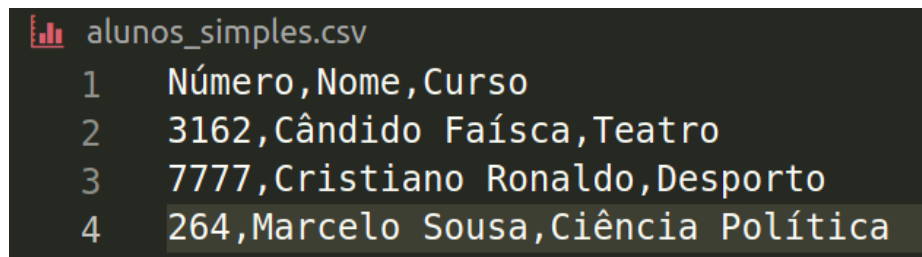
1 Ficheiros CSV com listas e funções de agregação

O enunciado do problema em caso é simples. Conseguir converter um ficheiro em formato CSV para o formato JSON.

Um dos maiores desafios é o facto do cabeçalho do ficheiro CSV pode conter **listas** com tamanho variável para os seus parâmetros e podem ser aplicadas **funções** a essas listas.

1.1 Conversor Simples de CSV para JSON

Apresentando o seguinte ficheiro `alunos_simples.csv`



```
alunos_simples.csv
1  Número, Nome, Curso
2  3162, Cândido Faísca, Teatro
3  7777, Cristiano Ronaldo, Desporto
4  264, Marcelo Sousa, Ciência Política
```

Fig. 1. Ficheiro Simples de CSV.

O programa lê a primeira linha do texto corresponde ao cabeçalho. De seguida descobre o nome dos parâmetros, sendo que os guarda em um *array de strings*.

A partir do número de elementos do array constroi um expressão de captura: $(.*?), (.*?), (.*?)$

Depois é construída a expressão de substituição dependendo do nome e do índice dos parâmetros:

```
{"Número": "\1", "Nome": "\2", "Curso": "\3"}
```

Desta forma temos a expressão de captura e a expressão de substituição basta apenas aplicar à string no formato CSV para a converter em JSON.

$$jsonString = re.sub("pattern", "substituion_pattern", "string") \quad (1)$$

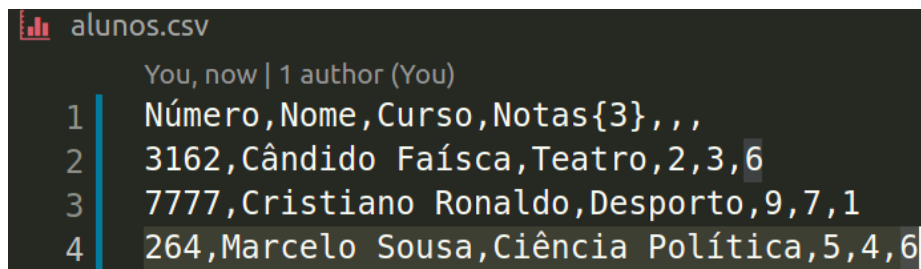
Sendo o output o seguinte:

```
[{"Número": "3162", "Nome": "Cândido Faísca", "Curso": "Teatro"},
{"Número": "7777", "Nome": "Cristiano Ronaldo", "Curso": "Desporto"},
{"Número": "264", "Nome": "Marcelo Sousa", "Curso": "Ciência Política"} ]
```

Com esta abordagem criamos um simples e eficaz conversor de CSV para JSON.

1.2 Conversor CSV com parâmetros de tamanho variavel

Foi nos apresentado o seguinte formato :



```
alunos.csv
You, now | 1 author (You)
1 Número, Nome, Curso, Notas{3}, , ,
2 3162, Cândido Faísca, Teatro, 2, 3, 6
3 7777, Cristiano Ronaldo, Desporto, 9, 7, 1
4 264, Marcelo Sousa, Ciência Política, 5, 4, 6
```

Fig. 2. Ficheiro de CSV com parâmetros variaveis.

Não muito diferente da aplicação anterior foi só necessário capturar parâmetros que podem ter um tamanho variavel e ajustar a expressão de captura

```
(.*?), (.*?), (.*?), (.*?), (.*?), (.*?)
```

Já para construir a expressão de substituição é necessário guardar todos os parametros numa estrutura de dados propria.

Para construir a expressão de substituição para tamanho variavel é necessário buscar o índice de começo e de fim à estrutura de dados.

```
{"Número": "\1", "Nome": "\2", "Curso": "\3", "Notas": [\4, \5, \6]}
```

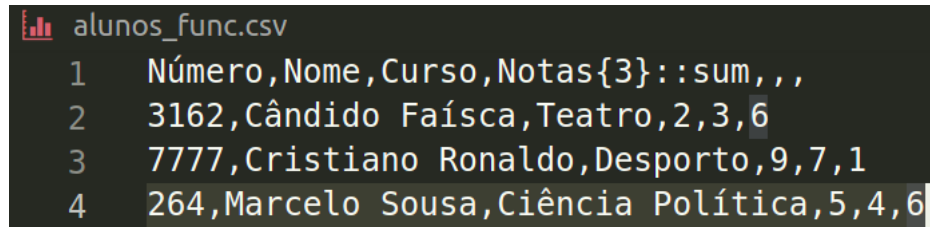
1.3 Conversor CSV com parâmetros de tamanho variavel entre 2 inteiros

Para o seguinte cabeçalho:

```
Número, Nome, Curso, Notas{1,3}, , ,
```

A estratégia adotada foi igual à apresentada anteriormente

1.4 Conversor CSV com funções parametrizadas



```
alunos_func.csv
1  Número, Nome, Curso, Notas{3}::sum,,,
2  3162, Cândido Faísca, Teatro, 2, 3, 6
3  7777, Cristiano Ronaldo, Desporto, 9, 7, 1
4  264, Marcelo Sousa, Ciência Política, 5, 4, 6
```

Fig. 3. Ficheiro CSV com funções associadas aos parâmetros.

Para poder lidar com este parâmetro extra foi necessário guardar na nossa estrutura de dados um **ENUM** chamado *behaviour*.

*Este behaviour pode ser atualmente de 3 tipos: **NO_BEHAVIOUR**, **SUM** ou **MEDIA**.*

(2)

Depois da operação de substituição ter sido realizada vamos iterar sobre os elementos da estrutura de dados para aplicar a função guardada em *behaviour*.

Tendo descoberto o novo valor realizamos uma nova substituição sobre a string em substituímos os dados anteriores pelos output da função através da seguinte função

```
jsonEntry = re.sub(
  fr'"{specialHeader.name}": \[.*?\]',
  fr'"{specialHeader.name}": {newnumber}',
  jsonEntry)
```

2 Testes

A aplicação tem uma variável **ENABLE_DEBUG** que por default é iniciada a **False** para não enviar para o default *stdout* as strings de captura que gera.

Para correr o programa no terminal:

```
python3 csv_2_json.py ficheiro.csv
```

```
pinto@pinto-S145 ~/Documents/2 Semestre/Processamento-de-Linguagens/TP01 master ± python3 csv_2_json.py alunos_simples.csv
Header ['Número', 'Nome', 'Curso']
Regex Expression (.?),(.?),(.*)
Substitution Expression {"Número": "\1", "Nome": "\2", "Curso": "\3"}
[{"Número": "3162", "Nome": "Cândido Faísca", "Curso": "Teatro"}, {"Número": "7777", "Nome": "Cristiano Ronaldo", "Curso": "Desporto"}, {"Número": "264", "Nome": "Marcelo Sousa", "Curso": "Ciência Política"} ]

pinto@pinto-S145 ~/Documents/2 Semestre/Processamento-de-Linguagens/TP01 master ± python3 csv_2_json.py alunos_variavel.csv
Header ['Número', 'Nome', 'Curso', 'Notas{3}']
Regex Expression (.?),(.?),(.?),(.?),(.?),(.*)
Substitution Expression {"Número": "\1", "Nome": "\2", "Curso": "\3", "Notas": [\4,\5,\6]}
[{"Número": "3162", "Nome": "Cândido Faísca", "Curso": "Teatro", "Notas": [2,3,6]}, {"Número": "7777", "Nome": "Cristiano Ronaldo", "Curso": "Desporto", "Notas": [9,7,1]}, {"Número": "264", "Nome": "Marcelo Sousa", "Curso": "Ciência Política", "Notas": [5,4,6]} ]

pinto@pinto-S145 ~/Documents/2 Semestre/Processamento-de-Linguagens/TP01 master ± python3 csv_2_json.py alunos_func.csv
Header ['Número', 'Nome', 'Curso', 'Notas{3}::sum']
Regex Expression (.?),(.?),(.?),(.?),(.?),(.*)
Substitution Expression {"Número": "\1", "Nome": "\2", "Curso": "\3", "Notas_sum": [\4,\5,\6]}
[{"Número": "3162", "Nome": "Cândido Faísca", "Curso": "Teatro", "Notas_sum": 11}, {"Número": "7777", "Nome": "Cristiano Ronaldo", "Curso": "Desporto", "Notas_sum": 17}, {"Número": "264", "Nome": "Marcelo Sousa", "Curso": "Ciência Política", "Notas_sum": 15} ]
```

O programa até pode executar sem levar parametros adicionais dessa forma vai ler todas linhas do **stdin**, podemos assim aproveitar do use de *pipes*.

```
pinto@pinto-S145 ~/Documents/2 Semestre/Processamento-de-Linguagens/TP01 master ± cat alunos_func.csv | python3 csv_2_json.py
Header ['Número', 'Nome', 'Curso', 'Notas{3}::sum']
Regex Expression (.?),(.?),(.?),(.?),(.?),(.*)
Substitution Expression {"Número": "\1", "Nome": "\2", "Curso": "\3", "Notas_sum": [\4,\5,\6]}
[{"Número": "3162", "Nome": "Cândido Faísca", "Curso": "Teatro", "Notas_sum": 11}, {"Número": "7777", "Nome": "Cristiano Ronaldo", "Curso": "Desporto", "Notas_sum": 17}, {"Número": "264", "Nome": "Marcelo Sousa", "Curso": "Ciência Política", "Notas_sum": 15} ]
```

3 Conclusão

A primeira abordagem para este problema foi de realizar uma operação de substituição sobre os dados usando a função

```
re.sub("pattern","substituion_pattern","string")
```

Concluimos que esta seria a melhor abordagem devido a ser a única ferramenta que tínhamos conhecimento. Numa posterior fase de desenvolvimento foi nos apresentado nas aulas a utilização de automatos finitos e o modulo *ply*.

Uma abordagem com essas ferramentas seria muito mais fácil de implementar e mais escalável. Ainda assim como já estava o trabalho quase terminado achamos melhor que iríamos continuar usando a função *sub*.

Este trabalho foi bastante útil para *meter-mos a mão na massa* e ficar confortáveis a usar expressões regulares.