Processamento de Linguagens (3º ano de Curso)

Trabalho Prático 1

Relatório de Desenvolvimento

André Martins	António Rodrigues	Rui Vieira
A89586	A89585	A89564

5 de abril de 2021

Resumo

Este relatório é relativo ao primeiro Trabalho Prático da Unidade Curricular de Processamento de Linguagens.
O enunciado correspondente ao nosso grupo de trabalho é um conversor genérico de CSV para JSON.

Conteúdo

Capítulo 1

Introdução

Supervisor: Pedro Rangel Henriques

1.1 Conversor Genérico de CSV para JSON

Área: Processamento de Linguagens

A equipa docente da Unidade Curricular de Processamento de Linguagens apresentou cinco enunciados correspondentes ao primeiro Trabalho Prático, sendo que o critério para a escolha do respetivo enunciado a cada grupo era dado pela fórmula Enunciado = (NumeroGrupo % 5) + 1. Como o nosso grupo é o 23, o enunciado que nos correspondia era o quarto. Tal como dito anteriormente, o objetivo final é apresentar um conversor de um qualquer ficheiro gravado em formato CSV para o formato JSON. É necessário e importante ter conhecimento para a conversão que a primeira linha do CSV dado funciona como um cabeçalho que descodifica a que correspondem os valores que vêm nas linhas seguintes. Teremos de ter em conta que o dataset poderá ter listas aninhadas em algumas células. Nesse caso o cabeçalho terá um asterisco '*' a seguir ao nome do respetivo campo. Se nada mais for colocado o conversor deverá converter cada valor dessa coluna numa lista em JSON. A seguir ao asterisco pode haver uma função de agregação: sum, avg, max, min. Aí o conversor terá de aplicar a operação de fold respetiva sobre a lista e produzir o JSON de acordo.

Estrutura do Relatório

No Capítulo 2 apresentamos a proposta dos docentes para o problema a solucionar, bem como, falamos dos conceitos básicos necessários para perceber o texto sendo estes explicados com mais detalhe. No Apêndice A apresenta-se a implementação e no Capítulo 3 é explicada a lógica por trás de cada função. No Capítulo 4 são apresentados os testes à solução proposta por nós para o problema.

Por fim, o Capítulo 5 apresenta as conclusões, discutindo-se os resultados obtidos.

Capítulo 2

Análise e Especificação

2.1 A Proposta

Como se vê na Figura 2.1 o pretendido com este trabalho é implementar um conversor que,e ao ler um ficheiro CSV processa a informação, transformando e escrevendo-a num ficheiro de formato JSON.

> ### 4 Conversor genérico de CSV para JSON
>
> Neste enunciado pretende-se fazer um conversor de um qualquer ficheiro gravado em formato CSV (*Comma Separated Values*, original e tipicamente usado para descarregar uma Folha de Cálculo num ficheiro de texto) para o formato JSON[9](um formato textual neutro e muito simples, baseado no conceito de um conjunto de pares { "campo": "valor"}, concorrente do XML enquanto sistema de exportação/transferência de dados entre aplicações para assegurar a interoperabilidade).
>
> Para poder realizar a conversão pretendida, é importante saber que a primeira linha do CSV dado funciona como cabeçalho que descodifica a que correspondem os valores que vêm nas linhas seguintes. Até aqui nada de novo . . . , mas é claro que leva mais uns ingredientes.
>
> O dataset dado poderá ter listas aninhadas nalgumas células. Mas nesse caso o cabeçalho terá um asterisco '*' a seguir ao nome do respetivo campo. Se nada mais for colocado o conversor deverá converter cada valor dessa coluna numa lista em JSON.
>
> Mas a seguir ao asterisco pode haver uma função de agregação: `sum, avg, max, min`. Aí o conversor terá de aplicar a operação de fold respetiva sobre a lista e produzir o JSON de acordo.

Figura 2.1: Enunciado proposto pela equipa docente

2.2 Conceitos básicos do enunciado

2.2.1 Conceitos básicos

- CSV
- JSON
- Python
- Expressões regulares

- Sum
- Avg
- Max
- Min

2.2.2 Definições dos conceitos básicos

Seguem-se as definições fundamentais para se perceberem as ideias defendidas a seguir:

CSV Comma Separated Values, original e tipicamente usado para descarregar uma Folha de Cálculo num ficheiro de texto;

JSON formato textual neutro e muito simples, baseado no conceito de um conjunto de pares { "campo": "valor"};

Python linguagem de programação de alto nível, interpretada por script, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte;

Expressões Regulares forma concisa e flexível de identificar cadeias de caracteres de interesse, como caracteres particulares, palavras ou padrões de caracteres;

Sum somatório de um conjunto de dados;

Avg média de um conjunto de dados;

Max elemento máximo de um conjunto de dados;

Min elemento mínimo de um conjunto de dados.

Capítulo 3

Explicação das funções usadas

isNumber(dados) Função que, recorrendo ao uso de um search, analisa se o campo é, ou não, um número.

isListaTruncada(dados) Função que, recorrendo ao uso de um search, analisa se o campo é, ou não, uma lista truncada.

calculaAvg(dados, campo, separator_lista) Função que recebe três strings. A primeira corresponde aos dados da lista truncada (dados), a segunda é o nome do campo (campo) a que diz respeito e a terceira indica o separador da lista (separator_lista). Usando a função split, separa-se a string dos dados pelo separator_lista armazenando a informação numa lista de float. Usando a função sum e len aplicadas à anterior lista calculamos a média dos dados. Usando a função sub substituimos no argumento campo o "*"pelo "_", retornando, por fim, o argumento campo, seguido pela variável calculada "media"em formato de String.

calculaSum(dados, campo, separator_lista) Função que recebe três strings. A primeira corresponde aos dados da lista truncada (dados), a segunda é o nome do campo (campo) a que diz respeito e a terceira indica o separador da lista (separator_lista). Usando a função split, separa-se a string dos dados pelo separator_lista armazenando a informação numa lista de float. Usando a função sum aplicada à anterior lista calculamos o somatório dos dados. Usando a função sub substituimos no argumento campo o "*"pelo "_", retornando, por fim, o argumento campo, seguido pela variável calculada "soma"em formato de String.

calculaMax(dados, campo, separator_lista) Função que recebe três strings. A primeira corresponde aos dados da lista truncada (dados), a segunda é o nome do campo (campo) a que diz respeito e a terceira indica o separador da lista (separator_lista). Usando a função split, separa-se a string dos dados pelo separator_lista armazenando a informação numa lista de float. Usando a função max aplicada à anterior lista calculamos o máximo dos dados. Usando a função sub substituimos no argumento campo o "*"pelo "_", retornando, por fim, o argumento campo, seguido pela variável calculada "maximo"em formato de String.

calculaMin(dados, campo, separator_lista) Função que recebe três strings. A primeira corresponde aos dados da lista truncada (dados), a segunda é o nome do campo (campo) a que diz respeito e a terceira indica o separador da lista (separator_lista). Usando a função split, separa-se a string dos dados pelo separator_lista armazenando a informação numa lista de float. Usando a função min aplicada à anterior lista calculamos o mínimo dos dados. Usando a função sub substituimos no argumento campo o "*"pelo "_", retornando, por fim, o argumento campo, seguido pela variável calculada "minimo"em formato de String.

conversor(csv, fileOutput, separator, separator_lista) Função responsável por ler os dados do ficheiro CSV e escrevê-los no ficheiro JSON. O ficheiro CSV é aberto com a função pré-definida open, com o modo de codificação binária "UTF-8"ativo. É aberto um descritor de ficheiro em modo de escrita para o ficheiro JSON com o nome passado no segundo input da função principal, caso este ficheiro não exista é criado um novo. É lida a primeira linha do ficheiro CSV com a função pré-definida readLine, que vai ignorar qualquer espaço no início e fim desta, devido à função pré-definida strip. É separada e colocada numa lista pela função split, com o separador definido na função principal (terceiro input). Esta lista vai ser guardada numa variável campos responsável por indicar a que correspondem os valores que vêm nas linhas seguintes. De seguida cada linha do ficheiro é lida individualmente e é feito o seu split pelo separator passado, sendo, também, ignorados quaisquer espaços existentes no início ou fim da mesma. Os dados são guardados numa variável valores, que é uma lista de String. Verifica-se se a primeira linha do ficheiro é truncada ou não. Em caso negativo realiza-se outra verificação para confirmar se existem campos que sejam número e, posto isto, a informação é processada e adicionada à variável output para no final ser escrita no ficheiro JSON. Em caso afirmativo são usadas Regex para procurar por listas aninhadas no dataset. Quando essas listas são encontradas pelas Regex as respetivas funções definidas por nós são chamadas e a informação é processada tendo em conta isso e depois adicionada à variável output. Por fim, de forma a terminar o processo, a informação contida na variável output é escrita no ficheiro JSON pretendido.

Recorrendo ao funcionamento do programa como um script, de seguida pedem-se três inputs ao utilizador: - O nome do ficheiro CSV a ler; - O nome do ficheiro JSON para onde a informação vai ser escrita, caso o ficheiro JSON não exista, então é criado um com o nome dado no segundo input; - O caracter que separa os campos de dados no ficheiro CSV; Após a receção destes parâmetros, é chamada a função conversor com os mesmos, sendo que ela é responsável por executar a transformação dos dados entre ficheiros. No final é impressa uma mensagem a notificar que a conversão foi bem executada, e é criado o ficheiro JSON correspondente.

Capítulo 4

Testes

4.1 Testes realizados e Resultados

Mostram-se a seguir alguns testes feitos e os respectivos resultados obtidos que serão analisados e comentados de seguida no Capítulo 5. Em título de nota informamos que o ficheiro Emd resulta de um CSV utilizado nas aulas práticas da Unidade Curricular.

notas.csv

```
numero;notas*max;notas*min;curso;notas*avg;nome;notas*sum;notas*
A71823;(12,14,15,18);(12,14,15,18);MIEI;(12,14,15,18);Ana Maria;(12,14,15,18);(12,14,15,18)
A89765;(11,16,13);(11,16,13);LCC;(11,16,13);Joao Martins;(11,16,13);(11,16,13)
A54321;(17,18);(17,18);MIEFIS;(17,18);Paulo Correia;(17,18);(17,18)
```

Figura 4.1: CSV das Notas

notas.json > ...


```
[
{
"numero": "A71823",
"notas_max": 18.0,
"notas_min": 12.0,
"curso": "MIEI",
"notas_avg": 14.75,
"nome": "Ana Maria",
"notas_sum": 59.0,
"notas": [12,14,15,18]
},
{
"numero": "A89765",
"notas_max": 16.0,
"notas_min": 11.0,
"curso": "LCC",
"notas_avg": 13.333333333333334,
"nome": "Joao Martins",
"notas_sum": 40.0,
"notas": [11,16,13]
},
{
"numero": "A54321",
"notas_max": 18.0,
"notas_min": 17.0,
"curso": "MIEFIS",
"notas_avg": 17.5,
"nome": "Paulo Correia",
"notas_sum": 35.0,
"notas": [17,18]
}
]
```


Figura 4.2: JSON das Notas

emd.csv

```
_id,index,dataEMD,nome/primeiro,nome/último,idade,género,morada,modalidade,clube,email,federado,resultado
6045074cd77860ac9483d34e,0,2020-02-25,Delgado,Gay,28,F,Gloucester,BTT,ACRroriz,delgado.gay@acrroriz.biz,true,true
6045074ca6adebd591b5d239,1,2019-07-31,Foreman,Prince,34,M,Forestburg,Ciclismo,ACDRcrespos,foreman.prince@acdrcrespos.org,false
6045074c221e2fdf430e9ef0,2,2021-01-06,Cheryl,Berger,21,M,Umapine,Basquetebol,Vitoria,cheryl.berger@vitoria.biz,false,true
6045074c529cbdce549d3923,3,2020-11-19,Graves,Goff,29,F,Babb,Andebol,AVCfamalicão,graves.goff@avcfamalicão.co.uk,false,false
6045074c3319a0f9e79aad87,4,2019-09-01,Mckay,Bolton,29,F,Chilton,Futebol,ACDRcrespos,mckay.bolton@acdrcrespos.me,false,false
```

Figura 4.3: CSV do Emd

emd.json > {} 0


```
[
{
"_id": "6045074cd77860ac9483d34e",
"index": 0,
"dataEMD": "2020-02-25",
"nome/primeiro": "Delgado",
"nome/ltimo": "Gay",
"idade": 28,
"g nero": "F",
"morada": "Gloucester",
"modalidade": "BTT",
"clube": "ACRroriz",
"email": "delgado.gay@acrroriz.biz",
"federado": "true",
"resultado": "true"
},
{
"_id": "6045074ca6adebd591b5d239",
"index": 1,
"dataEMD": "2019-07-31",
"nome/primeiro": "Foreman",
"nome/ltimo": "Prince",
"idade": 34,
"g nero": "M",
"morada": "Forestburg",
"modalidade": "Ciclismo",
"clube": "ACDRcrespos",
"email": "foreman.prince@acdrcrespos.org",
"federado": "false",
"resultado": "true"
},
```


Figura 4.4: JSON do Emd

Capítulo 5

Conclusão

Após a conclusão de toda a implementação consideramos que este desafio se demonstrou um ótimo quebra-cabeças que nos permitiu conhecer os formatos textuais CSV e JSON e nos levou a desenvolver o nosso conhecimento e prática com expressões regulares bem como com a linguagem de programação Python, que nunca antes tivera sido usada por nós. Em retrospetiva, consentimos que a nossa solução para este problema se demonstra capaz e eficaz e que nos apresentamos aptos a superar o desafio proposto.

Apêndice A

Código do Programa

Lista-se a seguir o código do programa que foi desenvolvido.

```
import re

def isNumber(dados):
    return re.search(r'^[0-9]+(\.[0-9]+)?$', dados)

def isListaTruncada(dados):
    return re.search(r'\*', dados)

def calculaAvg(dados, campo, separator_lista):
    lista = re.split(separator_lista, dados)
    for i in range(len(lista)):
        lista[i] = float(lista[i])
    media = sum(lista)/len(lista)
    campo = re.sub(r'\*', r'_', campo)
    return ("\"" + campo + "\": " + str(media))

def calculaSum(dados, campo, separator_lista):
    lista = re.split(separator_lista, dados)
    for i in range(len(lista)):
        lista[i] = float(lista[i])
    soma = sum(lista)
    campo = re.sub(r'\*', r'_', campo)
    return ("\"" + campo + "\": " + str(soma))

def calculaMax(dados, campo, separator_lista):
    lista = re.split(separator_lista, dados)
    for i in range(len(lista)):
        lista[i] = float(lista[i])
    maximo = max(lista)
    campo = re.sub(r'\*', r'_', campo)
    return ("\"" + campo + "\": " + str(maximo))

def calculaMin(dados, campo, separator_lista):
```

```
    lista = re.split(separator_lista, dados)
    for i in range(len(lista)):
        lista[i] = float(lista[i])
    minimo = min(lista)
    campo = re.sub(r'\*', r'_', campo)
    return ("\"" + campo + "\": " + str(minimo))

def conversor(csv, fileOutput, separator, separator_lista):
    file = open(csv,encoding="utf8")
    fileOutput = open(fileOutput, 'w')
    first_line = file.readline()
    campos = re.split(separator, first_line.strip())

    output = ""
    output += "[\n"

    for line in file:
        output += ("{\n")
        valores = re.split(separator, line.strip())
        for i in range(len(campos)):
            if not isListaTruncada(campos[i]):
                if isNumber(valores[i]):
                    output += ("\"" + campos[i] + "\": " + valores[i] + ",\n")
                else:
                    output += ("\"" + campos[i] + "\": \"" + valores[i] + "\",\n")
            else:
                valor = re.sub(r"\(", r"", valores[i])
                valor = re.sub(r"\)", r"", valor)

                if re.search(r'avg', campos[i]):
                    output += calculaAvg(valor, campos[i], separator_lista)
                    output += ",\n"

                elif re.search(r'sum', campos[i]):
                    output += calculaSum(valor, campos[i], separator_lista)
                    output += ",\n"

                elif re.search(r'max', campos[i]):
                    output += calculaMax(valor, campos[i], separator_lista)
                    output += ",\n"

                elif re.search(r'min', campos[i]):
                    output += calculaMin(valor, campos[i], separator_lista)
                    output += ",\n"

                else:
                    lista = re.split(separator_lista, valor)
                    campo = re.sub(r'\*', r"", campos[i])
```

```
                    output += ("\"" + campo + "\": ")
                    output += "["
                    for index in range(len(lista)):
                        output += str(lista[index])
                        output += ","
                    output += "]"
                    output = re.sub(r',]', r']', output)
                    output += (",\n")
        output += ("},\n")

    output += ("]")

    output = re.sub(r",\n}", r"\n}", output)
    output = re.sub(r"},\n]", r"}\n]", output)
    fileOutput.write(output)

csv = input('Insira o ficheiro CSV que pretende ler: ')
json = input('Insira o nome do ficheiro JSON: ')
separator = input('CSV separado por:\n1) ;\n2) ,\n')
if separator == "1":
    conversor(csv, json, ";", ",")
    print('Conversão realizada com sucesso!')
elif separator == "2":
    conversor(csv, json, ",", ";")
    print('Conversão realizada com sucesso!')
else :
    print("Opção inválida")
```