



Universidade do Minho
Departamento de Informática

Processamento de Linguagens TP1
Grupo 42

27 de Março, 2021



Alexandre Flores
(a93220)



Mariana Rodrigues
(a93294)



Matilde Bravo
(a93246)

Conteúdo

| | | |
|----------|--|----------|
| 1 | Introdução | 3 |
| 2 | Descrição do Enunciado Proposto | 4 |
| 2.1 | Exemplo | 4 |
| 3 | Implementação | 6 |
| 3.1 | Leitura do Header | 6 |
| 3.2 | Leitura e criação do json | 7 |
| 3.3 | Utilização | 7 |
| 3.4 | Exemplos de Utilização | 8 |
| 3.5 | Conclusão | 10 |

Capítulo 1

Introdução

No âmbito da unidade curricular de *Processamento de linguagens* foram propostos diversos enunciados, de entre os quais selecionamos o primeiro projeto. Este corresponde ao tema *Ficheiros CSV com listas e funções de agregação*.

O objetivo principal deste projeto é converter ficheiros *CSV* em ficheiros *JSON*. Para o fazermos, foi necessária uma adequada análise e interpretação programáticas dos dados do *CSV*. Este processo, bem como o da escrita geral de um ficheiro *JSON* válido, é descrito ao longo deste documento.

Capítulo 2

Descrição do Enunciado Proposto

O enunciado propõe a criação de um conversor de ficheiros **CSV** (Comma Separated Values) para o formato **JSON**. Para esse efeito, é necessária a existência de um cabeçalho nos ficheiros a converter, por forma a identificar os campos ou colunas a serem lidos nas linhas posteriores. Além do nome do campo, informação que costuma constar num ficheiro *CSV* tradicional, o header poderá conter mais informação à cerca do campo. São suportadas listas, as quais poderão ser identificadas no cabeçalho, cujo tamanho é especificado entre chavetas. Por exemplo, se no *header* constar 'nome_campo{N}', significa que para esse campo terão que ser lidos obrigatoriamente N valores. Por outro lado, também é possível indicar um intervalo de valores: 'nome_campo{N,M}', o que permite a existência de um número variável de colunas (entre N e M). Para além das listas, é possível ter funções de agregação que serão aplicadas às listas de valores e o resultado da sua aplicação será o valor apresentado no JSON ao invés da própria lista. Além disso, ao nome do campo sera acrescentado, como sufixo, o nome da função aplicada.

2.1 Exemplo

O seguinte ficheiro CSV de input, deverá gerar o JSON que o segue.

```
Número,Nome,Curso,Notas{3}::media,Desporto{1,3},,,,
a93232,Joana Epifânio,Línguas Aplicadas,12,13,14,Paddle,,
a92123,Catarina Belo,Relações Internacionais,17,12,20,Paddle,,
a93212,Marcelo Feio,Engenharia Biomédica,18,13,19,Futebol,Basquetebol,
a91234,Pedro Castilho,Psicologia,18,19,19,Voleibol,Futebol,Ténis

[
  {
    "Número": "a93232",
    "Nome": "Joana Epifânio",
    "Curso": "Línguas Aplicadas",
    "Notas_media": 19,
    "Desporto": [
      "Paddle"
    ]
  },
  {
    "Número": "a92123",
    "Nome": "Catarina Belo",
    "Curso": "Relações Internacionais",
    "Notas_media": 24.33333333333332,
    "Desporto": [
      "Paddle"
    ]
  }
]
```

```
},
{
  "Número": "a93212",
  "Nome": "Marcelo Feio",
  "Curso": "Engenharia Biomédica",
  "Notas_media": 22.666666666666668,
  "Desporto": [
    "Futebol",
    "Basquetebol"
  ],
  {
    "Número": "a91234",
    "Nome": "Pedro Castilho",
    "Curso": "Psicologia",
    "Notas_media": 24.666666666666668,
    "Desporto": [
      "Voleibol",
      "Futebol",
      "Ténis"
    ]
  }
]
```

Capítulo 3

Implementação

3.1 Leitura do Header

Estrutura de Dados

Uma vez que o cabeçalho contém informação relevante para a forma como as restantes linhas vão ser lidas, é fulcral armazenar esta informação de uma forma simples e que possa ser consultada aquando da leitura do resto do CSV. Para este fim, foi criada a classe **Field**.

```
class Field:  
    def __init__(self, name, minimum = 1, maximum = 1, agg_fun = lambda x:x):  
        self.name = name  
        self.min = int(minimum)  
        self.max = int(maximum)  
        self.agg_fun = agg_fun  
  
    def __str__(self):  
        return "name: %s , min: %s, max: %s, agg_fun: %s" %  
            (self.name, self.min, self.max, self.agg_fun.__name__)
```

Objetos desta classe são criados aquando da leitura do cabeçalho do CSV. O nome das colunas é necessário, pelo que se trata de um parâmetro obrigatório para instanciar esta classe. Opcionalmente, e como mencionado na secção anterior, poderão ser especificados um intervalo de valores e uma função de agregação. Caso não o sejam, um valor de 1 para o mínimo e o máximo e a função identidade serão assumidos.

Tokenizer

Por forma a facilitar o processamento do header, foi criada um *tokenizer*, tendo em mente o que foi lecionado nas aulas teóricas. A expressão regular para identificar o nome de uma coluna está guardada na variável **str_regex** e a expressão regular que identifica um intervalo de valores na variável **rng_regex**. A variável **separator** contém o separador especificado pelo utilizador na leitura do CSV. As tokens do **header** são: função de agregação, representada por AGG_FUNCTION, intervalo, representado por RANGE e string, representada por STR. Através da lista de tuplos constituídos por nome da token e respetiva expressão regular para a identificar, **token_specification**, é possível construir um *regex* que captura cada uma destas. As expressões regulares associadas a cada token são agregadas numa só *string*, através de uma disjunção. Além disso, são utilizados *named groups* para ser possível aceder mais facilmente, através de um dicionário, às *tokens* encontradas. É de salientar que, apesar de ter sido proposto a utilização de uma função de agregação no caso de se tratar de uma lista, optamos por expandir o conceito de função de agregação para uma qualquer função que possa ser aplicada a um valor e não apenas a uma lista .

```

str_regex = r'[a-zA-Z_]+[^'+separator+r'\n]*'
rng_regex = r"(\w+)\{(\d+)(?:,(\d+))?\}"

token_specification = [
    ('AGG_FUNCTION', r'(%s|%)::(\w+)' % (str_regex, rng_regex)),
    ('RANGE', rng_regex),
    ('STR', str_regex),
]

tok_regex = '|'.join((r'(?P<%s>%s)' % pair for pair in token_specification))

```

A função principal de *parsing* do **header** é `parse_header`. O dicionário que resulta da procura é passado à função `create_field` que será responsável por ver quais as *tokens* presentes e criar o objeto `Field` com a informação lida.

```

def parse_header(line):
    r = re.finditer(tok_regex, line)
    for i in r:
        dic = i.groupdict()
        create_field(dic)

```

A função `create_field` procura o valor do dicionário que não for nulo, por forma a determinar a *token* descoberta. De seguida, são utilizados os valores capturados pelos vários grupos.

3.2 Leitura e criação do json

Uma vez lida e armazenada toda a informação relevante do *cabeçalho*, procede-se à leitura das linhas seguintes.

Com o intuito de melhorar a organização, decidimos organizar este processo em duas funções:

```

def parse_Field(line, minimo, maximo)
def csv_to_json(f)

```

A função `csv_to_json(f)` encontra-se responsável pelo *parsing* de todas as restantes linhas. Por outro lado, a função `parse_Field` encontra-se responsável apenas por formatar uma dada linha para o formato devido.

Uma vez que, cada linha se encontra no formato CSV conseguimos separar cada um dos elementos da linha, através da seguinte expressão de **regex**:

```
r = re.compile(r'[^'+separator+'\n]*'+separator+'?')
```

Onde **separator** é uma variável que contém o separador atual do CSV.

Sendo assim, é simples separar e obter todos os campos dessa dada linha:

```
list_words = r.findall(line)
```

Tendo toda esta informação, resta apenas percorrer todos os objectos por nós criados da class `Field`, e dependendo da informação que cada objeto contiver, ir recolhendo e manipulando os elementos presentes em `list_words`.

3.3 Utilização

O programa lê os ficheiros CSV do *Standard Input* e envia o resultado, o ficheiro JSON, para o *Standard Output*. Por defeito o programa assume a vírgula como o separador utilizado no ficheiro CSV. O utilizador pode especificar um separador com recurso à flag -F. Seguem-se alguns exemplos de utilização:

```

cat input.csv | python main.py -F ;
cat input.csv | python main.py
python main.py < ficheiro.csv > output.json

```

3.4 Exemplos de Utilização

As seguintes *screenshots* ilustram a utilização do programa.

- Ficheiro de teste disponibilizado

| | File: emd.csv |
|----|---|
| 1 | _id, index, dataEMD, nome/primeiro, nome/ultimo, idade, género, morada, modalidade, clube, email, federado, resultado |
| 2 | 6045074cd77860ac9483d34e, 0, 2020-02-25, Delgado, Gay, 28, F, Gloucester, BTT, ACRroriz, delgado.gay@acrrioriz.biz, true, true |
| 3 | 6045074ca6adebd591b5d239, 1, 2019-07-31, Foreman, Prince, 34, M, Forestburg, Ciclismo, ACDRcrespos, foreman.prince@acdrcrespos.org, false, true |
| 4 | 6045074c221e2fdf430e9ef0, 2, 2021-01-06, Cheryl, Berger, 21, M, Umapine, Basquetebol, Vitoria, cheryl.berger@vitoria.biz, false, true |
| 5 | 6045074c529cbdc549d3923, 3, 2020-11-19, Graves, Goff, 29, F, Bab, Andebol, AVFciamalicião, graves.goff@avfciamalicião.co.uk, false, false |
| 6 | 6045074c3319a0f9e79a8d87, 4, 2019-09-01, McKay, Bolton, 29, F, Chilton, Futebol, ACDRcrespos, mckay.bolton@acdrcrespos.me, false, false |
| 7 | 6045074c222607e7520fd24, 5, 2019-10-07, Marla, Kelley, 22, M, Clarence, Atletismo, AmigosMontanha, marla.kelley@amigosmontanha.tv, false, false |
| 8 | 6045074c344b052551309595, 6, 2020-12-11, Merrill, Maddox, 34, M, Roderfield, Equitação, ACRroriz, merrill.maddox@acrrioriz.com, false, true |
| 9 | 6045074c6619e6ba7eca2f8d, 7, 2019-03-12, Nikki, Calderon, 22, F, Brether, Patinagem, EDViana, nikki.calderon@edviana.name, true, false |
| 10 | 6045074c0ae66598285f526, 8, 2020-12-09, Lucia, Bright, 35, F, Onton, Orientação, GDGoma, lucia.bright@gdgoma.info, true, false |
| 11 | 6045074c1e1efef942191e9f9, 9, 2020-01-26, Faith, Wells, 31, F, Grantville, Basquetebol, EDViana, faith.wells@edviana.co, true, true |
| 12 | 6045074cb789f82702d32997, 10, 2019-03-29, Milagros, Osborn, 22, M, Junio, Patinagem, GDGoma, milagros.osborn@gdgoma.io, false, false |
| 13 | 6045074cc8eaab6bbbed60fc9, 11, 2019-09-08, Valentine, Sellers, 27, F, Odessa, Patinagem, EDViana, valentine.sellers@edviana.net, true, true |
| 14 | 6045074c6c87caf7f00f0050, 12, 2019-05-10, Martha, Hyde, 26, M, Dupuyer, Triatlo, Vitoria, martha.hyde@vitoria.biz, false, false |
| 15 | 6045074cd3867699e6852a, 13, 2019-04-22, Beck, Stevenson, 35, F, Barclay, Dança, SCBraga, beck.stevenson@scbraga.org, true, false |
| 16 | 6045074ca84d18be9281b55a, 14, 2019-05-05, Lester, Strong, 21, M, Freetown, Patinagem, GDGoma, lester.strong@gdgoma.biz, false, false |
| 17 | 6045074cbea2f209cb62712, 15, 2020-03-19, Sharon, Bradley, 30, M, Hiserville, Futebol, ABCBraga, sharon.bradley@avfciamalicião.co.uk, true, false |
| 18 | 6045074ceaa529d03ad4ee0c2, 16, 2019-07-30, Waters, Dale, 32, M, Bakerville, Andebol, ABCBraga, waters.dale@bcbbraga.me, false, false |
| 19 | 6045074cc38fa636edf15de0, 17, 2019-07-27, Rebekah, Mayer, 32, F, Woodlake, Atletismo, ABCBraga, rebekah.mayer@bcbbraga.tv, false, false |
| 20 | 6045074c8aa7f30b690566eb, 18, 2020-04-19, Isabella, Howell, 28, M, Hanover, Basquetebol, AVFciamalicião, isabella.howell@avfciamalicião.com, false, false |
| 21 | 6045074c112d6ee3d5dde27a, 19, 2019-12-28, Lela, Barnes, 31, F, Sulking, Karaté, EDViana, lela.barnes@edviana.name, false, true |

Figura 3.1: emd.csv

| | File: emd.json |
|----|--|
| 1 | { |
| 2 | { |
| 3 | "_id": "6045074cd77860ac9483d34e", |
| 4 | "index": "0", |
| 5 | "dataEMD": "2020-02-25", |
| 6 | "nome/primeiro": "Delgado", |
| 7 | "nome/ultimo": "Gay", |
| 8 | "idade": "28", |
| 9 | "género": "F", |
| 10 | "morada": "Gloucester", |
| 11 | "modalidade": "BTT", |
| 12 | "clube": "ACRroriz", |
| 13 | "email": "delgado.gay@acrrioriz.biz", |
| 14 | "federado": "true", |
| 15 | "resultado": "true" |
| 16 | }, |
| 17 | { |
| 18 | "_id": "6045074ca6adebd591b5d239", |
| 19 | "index": "1", |
| 20 | "dataEMD": "2019-07-31", |
| 21 | "nome/primeiro": "Foreman", |
| 22 | "nome/ultimo": "Prince", |
| 23 | "idade": "34", |
| 24 | "género": "M", |
| 25 | "morada": "Forestburg", |
| 26 | "modalidade": "Ciclismo", |
| 27 | "clube": "ACDRcrespos", |
| 28 | "email": "foreman.prince@acdrcrespos.org", |
| 29 | "federado": "false", |
| 30 | "resultado": "true" |
| 31 | }, |
| 32 | { |
| 33 | "_id": "6045074c221e2fdf430e9ef0", |
| 34 | "index": "2", |
| 35 | "dataEMD": "2021-01-06", |
| 36 | "nome/primeiro": "Cheryl", |
| 37 | "nome/ultimo": "Berger", |
| 38 | "idade": "21", |
| 39 | "género": "M", |
| 40 | "morada": "Umapine", |
| 41 | "modalidade": "Basquetebol", |
| 42 | "clube": "Vitoria", |
| 43 | "email": "cheryl.berger@vitoria.biz", |
| 44 | "federado": "false", |
| 45 | "resultado": "true" |
| 46 | }, |

Figura 3.2: emd.json

- Ficheiro com 5000 linhas de entrada

```
PL 💎 ls -l 5000\ Sales\ Records.csv 5000\ Sales\ Records.json
.rw-r--r-- 623k mari 28 Jul 2017 5000 Sales Records.csv
.rw-r--r-- 2.4M mari 19 Mar 16:37 5000 Sales Records.json
```

Figura 3.3: Tamanho do ficheiro de teste e do resultado obtido

| File: 5000 Sales Records.csv | | | | | | | | | | | | | |
|-----------------------------------|----------------------------------|-----------------|---------------|---------------------|---------------------|--------------|------------|------------|------------|---------------|---------------|--------------|-----------|
| Region | Country | Item Type | Sales Channel | Order Priority | Order Date | Order Status | Units Sold | Unit Price | Unit Total | Total Revenue | Total Cost | Total Profit | |
| Central America and the Caribbean | Antigua and Barbuda | Baby Food | Online | L-12/10/2013 | 057081544 | 1/11/2014 | 255 | 25.50 | 15,492 | 4409156 | 87,999.04 | 529149.72 | |
| Europe,Czech Republic | Beverages | Offline | L-03/12/2011 | 478051603 | 1/11/2014 | 21,209 | 31.79 | 678,186 | 15,182 | 62,748.00 | 111522.46 | 119488.38 | |
| Asia, North Korea | Cereal | Offline | L-5/3/2010 | 1000000000000000000 | 20/10/2010 | 20,705 | 11.71 | 238,450 | 18,058536 | 76,79877.44 | 111522.46 | 119488.38 | |
| Australia and Oceania | Asia, Sri Lanka | Clothes | Offline | L-11/12/2011 | 932766881 | 1/11/2014 | 48,817 | 55.50 | 2,716,200 | 4,027,720.00 | -1,311,520.00 | 111522.46 | 119488.38 |
| Middle East and North Africa | Morocco | Personal Care | Offline | L-11/3/2010 | 412852727 | 11/11/2013 | 2016 | 48.81 | 97 | 97,270.00 | 12,020.00 | 85,250.00 | |
| Australia and Oceania | Federated States of Micronesia | Clothes | Offline | L-1/28/2011 | 932766881 | 1/11/2014 | 8258 | 199.28 | 1,635 | 98,2434 | 24,99566 | 72,606467.52 | |
| Central America and the Caribbean | Bosnia and Herzegovina | Clothes | Offline | L-11/12/2011 | 932766881 | 1/11/2014 | 207 | 199.28 | 41 | 39,856.00 | 5,700.00 | 34,156.00 | |
| Middle East and North Africa | Algeria | Apparel | Offline | L-8/27/2015 | 570180403 | 1/11/2014 | 8841 | 25.35 | 223,845 | 56,911.44 | 167,933.56 | 111522.46 | 119488.38 |
| Sub-Saharan Africa | Ethiopia | Baby Food | Online | M-14/3/2015 | 192993512 | 1/11/2014 | 255 | 25.10 | 14,200 | 16,156926 | 14 | 941057.62 | |
| Middle East and North Africa | Turkey | Office Supplies | Offline | L-11/12/2011 | 205170103 | 20/11/2013 | 1,000 | 37.00 | 37,000 | 37,000.00 | 0.00 | 111522.46 | 119488.38 |
| Central America and the Caribbean | Yemen | Clothes | Offline | L-12/1/2012 | 1000000000000000000 | 1/11/2014 | 3764 | 69.31 | 25,124 | 95,744.00 | 111522.46 | 119488.38 | |
| Asia, Malaysia | Cereal | Offline | L-7/31/2016 | 333942162 | 8/25/2016 | 9765 | 20 | 117,114 | 30,880034 | 40,1143227 | 82,864815.56 | | |
| Central America and the Caribbean | Saint Lucia | Cosmetic | Offline | L-7/6/2015 | 795105983 | 7/10/2015 | 6786 | 45.70 | 305,263 | 33,300.00 | 56639.20 | 786,995.86 | |
| Central America and the Caribbean | Saint Vincent and the Grenadines | Clothes | Offline | L-11/28/2010 | 504315584 | 1/11/2014 | 201 | 199.28 | 40,200 | 158,42 | 100,903.00 | 104,2751.75 | |
| Middle East and North Africa | Lebanon | Meat | Offline | H-12/17/2015 | 616297950 | 1/11/2014 | 2016 | 369.00 | 741 | 24,894.00 | 7,138,000.00 | 7,120,900.00 | |
| Europe,Bulgaria | Office Supplies | Online | L-10/21/2019 | 2703230000000000000 | 1/11/2014 | 655 | 25.1 | 16,390 | 60,8481 | 57,927,124 | 111522.46 | 119488.38 | |
| North America,Mexico | Beverages | Online | C-3/3/2017 | 127374363 | 1/11/2014 | 1747 | 45 | 79,31 | 79,655.00 | 55378 | 1,27279.72 | | |
| Central America and the Caribbean | Argentina | Clothes | Offline | L-11/12/2011 | 932766881 | 1/11/2014 | 2016 | 25.50 | 508,256 | 70,450.00 | 437,806.00 | | |
| Middle East and North Africa | Algeria | Beverages | Offline | L-11/20/2019 | 993355010 | 1/11/2014 | 1718 | 47 | 81,31 | 79,513.00 | 54,84512 | 22,26903.88 | |

Figura 3.4: Amostra do ficheiro de teste

| | File: 5000 Sales Records.json | (END) |
|----|--|-------|
| 1 | [| 79953 |
| 2 | { | 79955 |
| 3 | "Region": "Central America and the Caribbean", | 79956 |
| 4 | "Country": "Antigua and Barbuda", | 79957 |
| 5 | "Item Type": "Baby Food", | 79958 |
| 6 | "Sales Channel": "Online", | 79959 |
| 7 | "Order Priority": "", | 79970 |
| 8 | "Order Date": "12/29/2013", | 79971 |
| 9 | "Order ID": "95708344", | 79972 |
| 10 | "Ship Date": "1/1/2014", | 79973 |
| 11 | "Units Sold": "552", | 79974 |
| 12 | "Unit Price": "255.28", | 79975 |
| 13 | "Unit Cost": "159.42", | 79976 |
| 14 | "Total Revenue": "140914.56", | 79977 |
| 15 | "Total Cost": "87998.84", | 79978 |
| 16 | "Total Profit": "52915.72" | 79980 |
| 17 | } | 79981 |
| 18 | { | 79982 |
| 19 | "Region": "Central America and the Caribbean", | 79983 |
| 20 | "Country": "Panama", | 79984 |
| 21 | "Item Type": "Books", | 79985 |
| 22 | "Sales Channel": "Offline", | 79986 |
| 23 | "Order Priority": "C", | 79987 |
| 24 | "Order Date": "7/7/2010", | 79988 |
| 25 | "Order ID": "301644504", | 79989 |
| 26 | "Ship Date": "7/26/2010", | 79990 |
| 27 | "Units Sold": "1", | 79991 |
| 28 | "Unit Price": "152.58", | 79992 |
| 29 | "Unit Cost": "97.44", | 79993 |
| 30 | "Total Revenue": "330640.86", | 79994 |
| 31 | "Total Cost": "211152.48", | 79995 |
| 32 | "Total Profit": "119488.38" | 79996 |
| 33 | } | 79997 |
| 34 | { | 79998 |
| 35 | "Region": "Europe", | 79999 |
| 36 | "Country": "Czech Republic", | 80000 |

Figura 3.5: Amostra do ficheiro obtido

- Ficheiro com funções associadas

```

File: test.csv

Número;Nome;Curso::hello;Notas[3]::media;Desporto[0,3]::len;Valor::int
a93232;Joana Epifânia;Línguas Aplicadas;12;13;14;;2
a92123;Catarina Belo;Relações Internacionais;17;12;20;Paddle;;2
a93212;Marcelo Feio;Engenharia Biomédica;18;13;19;Futebol;Basquetebol;;1900
a91234;Pedro Castilho;Psicologia;18;19;19;Voleibol;Futebol;Ténis;3

```

Figura 3.6: Amostra do ficheiro de teste

É de salientar que a função **hello** encontra-se definida do seguinte modo:

```

def hello(l):
    return "Hello ::" + l + ":: Hello"

PL ✘ cat test.csv | python main.py -F ';' | jq
[
  {
    "Número": "a93232",
    "Nome": "Joana Epifânia",
    "Curso_hello": "Hello::Línguas Aplicadas::Hello",
    "Notas_media": 19,
    "Desporto_len": 0,
    "Valor_int": 2
  },
  {
    "Número": "a92123",
    "Nome": "Catarina Belo",
    "Curso_hello": "Hello::Relações Internacionais::Hello",
    "Notas_media": 24.333333333333332,
    "Desporto_len": 1,
    "Valor_int": 2
  },
  {
    "Número": "a93212",
    "Nome": "Marcelo Feio",
    "Curso_hello": "Hello::Engenharia Biomédica::Hello",
    "Notas_media": 22.666666666666668,
    "Desporto_len": 2,
    "Valor_int": 1900
  },
  {
    "Número": "a91234",
    "Nome": "Pedro Castilho",
    "Curso_hello": "Hello::Psicologia::Hello",
    "Notas_media": 24.666666666666668,
    "Desporto_len": 3,
    "Valor_int": 3
  }
]

```

Figura 3.7: Amostra do ficheiro obtido

3.5 Conclusão

Com a realização deste projeto pudemos consolidar a matéria inicial lecionada nesta UC, exercitando a nossa capacidade de *parsing* de ficheiros utilizando expressões regulares. Por outro lado, observámos em primeira mão a grande versatilidade e as capacidades da linguagem *python* e do módulo *re*. Esta revelou-se crucial e muito eficaz ao longo do desenvolvimento deste projeto.

Finalmente, achamos o nosso trabalho satisfatório e consideramos que o nosso programa cumpre os objetivos estabelecidos no enunciado.