

# Métodos Probabilísticos para Engenharia Informática

## Guião PL04

Ano letivo 2023/2024

Turma P4

Ricardo Quintaneiro  
NMec: 110056

Rúben Garrido  
NMec: 107927

21 de dezembro de 2023

# Índice

<b>1</b>	<b>Introdução</b>	<b>2</b>
<b>2</b>	<b>Material fornecido</b>	<b>2</b>
<b>3</b>	<b>Desenvolvimento</b>	<b>3</b>
3.1	Hash function . . . . .	3
3.2	Opção 1 . . . . .	3
3.3	Opção 4 . . . . .	3
3.3.1	Estrutura de dados . . . . .	3
3.3.2	Assinaturas . . . . .	4
3.3.3	Pesquisa . . . . .	4
3.4	Opção 5 . . . . .	5
3.4.1	Estrutura de dados . . . . .	5
3.4.2	Assinaturas . . . . .	6
3.4.3	Pesquisa . . . . .	6

## 1 Introdução

No âmbito da Unidade Curricular de Métodos Probabilísticos para Engenharia Informática, foi-nos proposto realizar um trabalho prático relativo a Hash Functions, Bloom Filter e MinHash. Este consiste em desenvolver um script de consulta de filmes e géneros, que se relacionam entre si.

Continuar

## 2 Material fornecido

Foi-nos fornecido um ficheiro CSV (*Comma-Separated Values*), `movies.csv`, onde constam informações sobre mais de 50000 filmes.

Quanto à estrutura, na primeira coluna está presente o título, na segunda o ano e nas restantes os vários géneros associados ao filme.

## 3 Desenvolvimento

### 3.1 Hash function

Recorremos à função `muxDJB31MA` para efetuar o *hashing* em todas as ocasiões onde este é necessário.

Esta é uma modificação da função `DJB31MA` utilizada nas aulas, onde é usado um argumento `k` correspondente ao número de funções de *hash*. Isto permite reduzir a complexidade computacional, já que deixa de ser necessário invocar a função e percorrer a string para cada função de *hash*.

```
1 function aux = muxDJB31MA(chave, seed, k)
2 len = length(chave);
3 chave = double(chave);
4 h = seed;
5 aux = zeros(1, k);
6 for i=1:len
7     h = mod(31 * h + chave(i), 2^32 -1) ;
8 end
9 for i = 1:k
10    h = mod(31 * h + i, 2^32 -1) ;
11    aux(i) = h;
12 end
13 end
```

### 3.2 Opção 1

Filtrámos, através de um ciclo `for`, os demais géneros, de modo a obter um vetor de géneros únicos, ignorando os elementos do *cell array* que estão *missing*.

```
1 genres = {};
2 [x, y] = size(movies);
3
4 for i = 1:x
5     for j = 3:y
6         if ~ismissing(movies{i, j})
7             if ~ismember(convertCharsToStrings(movies{i, j}),
8                 genres)
9                 genres = [genres convertCharsToStrings(movies{i, j
10                     })];
11             end
12         end
13     end
14 end
15 end
```

### 3.3 Opção 4

#### 3.3.1 Estrutura de dados

Para criar as assinaturas necessárias à execução desta opção, é necessário criar uma estrutura de dados adequada, que permita posteriormente efetuar a comparação entre a string introduzida e os vários títulos.

Assim, para cada filme, são criados shingles a partir do título. Estes têm comprimento 2, uma vez que é o número que permite obter uma maior fiabilidade nos resultados, pelo facto de existir uma maior granularidade.

```

1 [Set_title, Nm] = createMovieTitleStructure(movies);
2
3 function [Set, Nm] = createMovieTitleStructure(movies)
4 % For each movie, get its title split in shingles.
5 Nm = length(movies);
6 Set = cell(Nm,1);
7
8 for n = 1:Nm % Lines (movies)
9     title = movies{n,1};
10    for i = 1:length(title)-1
11        Set{n} = [Set{n} convertCharsToStrings(title(i:i+1))];
12    end
13 end
14 end

```

### 3.3.2 Assinaturas

As assinaturas são geradas com a função `getSignatures`. Para cada conjunto do set, esta função faz uso do algoritmo MinHash: calcula a *hash* para cada elemento e guarda o menor valor obtido.

Foram utilizadas 100 funções de hash (ou seja,  $K = 100$ ), uma vez que consideramos ser um número adequado tendo em conta o tamanho do set em questão.

```

1 signatures_title = getSignatures(Set_title, Nm, 100);
2
3 function signatures = getSignatures(Set, Nm, K)
4 signatures = inf(Nm, K);
5
6 for n = 1:Nm
7     set_n = Set{n};
8     for i = 1:length(set_n)
9         key = num2str(set_n(i));
10        h_out = muxDJB31MA(key, 127, K);
11        signatures(n,:) = min(h_out, signatures(n,:));
12    end
13 end
14 end

```

### 3.3.3 Pesquisa

Para a string introduzida na consola do MATLAB, são repetidos os passos das secções 3.3.1 e 3.3.2, de modo a obter as suas assinaturas.

```

1 a = input("Insert a string: ", "s");
2 Set = cell(1,1);
3
4 for i = 1:length(a)-1
5     Set{1} = [Set{1} convertCharsToStrings(a(i:i+1))];
6 end
7

```

```

8 k = 100;
9 signatures = getSignatures(Set, 1, k);

```

De seguida, comparam-se ambas as matrizes de assinaturas para obter a distância (e consequentemente a similaridade) de Jaccard.

```

1 size_titles = size(signatures_title, 1);
2 similarities = zeros(1, size_titles);
3
4
5 for n = 1:size_titles
6     similarities(n) = 1 - (sum(signatures ~= signatures_title(n, :)
7                             ) / k);
7 end

```

Após obter o vetor de similaridades, é efetuada neste uma ordenação decrescente quanto à similaridade de Jaccard e aplicado um limite de 5 filmes, de acordo como é pedido no enunciado.

```

1 [sortedSimilarities, indices] = sort(similarities, 'descend');
2 topSimilarities = sortedSimilarities(1:5);
3 topTitles = Set_title(indices(1:5));

```

Por fim, e uma vez que temos um conjunto de shingles, efetuamos uma concatenação entre estes de modo a obter o título completo.

```

1 fprintf("Top 5 Similar Titles:\n");
2 concatenatedTitle = strings(length(topTitles{i}) + 1);
3 for i = 1:5
4     for j= 1:length(topTitles{i})
5         concatenatedTitle(i) = strcat(concatenatedTitle(i),
6                                       topTitles{i}{j}(1));
6     end
7     concatenatedTitle(i) = strcat(concatenatedTitle(i), topTitles{i}
8                                   {j}(2));
8     fprintf("\t%s - %f\n", concatenatedTitle(i), topSimilarities(i)
9           );
9 end

```

O output é, conforme esperado:

```

1 Insert a string:
2 Love
3
4 Top 5 Similar Titles:
5 Love - 1.000000
6 Lover - 0.690000
7 Love65 - 0.530000
8 Loveling - 0.520000
9 Lovelife - 0.520000

```

## 3.4 Opção 5

### 3.4.1 Estrutura de dados

A estrutura de dados é semelhante à encontrada para a opção 4 (ver secção 3.3.2). No entanto, ao invés de criar *shingles*, o set inclui os conjuntos de géneros para cada filme.

```

1 [Set_genre, Nm] = createMovieGenreStructure(movies);
2
3 function [Set, Nm] = createMovieGenreStructure(movies)
4 % For each movie, get its genres
5 Nm = length(movies);
6 Set = cell(Nm,1);
7
8 for n = 1:Nm % Lines (movies)
9     for g = 3:12 % Columns (genres)
10         if ~ismissing(movies{n, g})
11             Set{n} = [Set{n} convertCharsToStrings(movies{n,g})];
12         end
13     end
14 end
15 end

```

### 3.4.2 Assinaturas

O conjunto de assinaturas para esta opção foi obtido da mesma forma que a opção 4, utilizando a *hash function* (ver secção 3.3.2).

```

1 signatures_genre = getSignatures(Set_genre, Nm, 100);

```

### 3.4.3 Pesquisa

É pedida uma string de géneros no formato CSV pelo input do MATLAB, ao que é efetuado um split por vírgula, de modo a obter os diversos géneros que nesta se encontram.

```

1 a = input("Select one or more genres (separated by ','): ", "s");
2 a = strsplit(a, ',');

```

De seguida, é criada uma matriz de assinaturas para os vários géneros, tal como na opção 4.

```

1 Set = strings(length(a),1);
2 for i= 1:length(a)
3     Set(i,1) = convertCharsToStrings(a(i));
4 end
5 b = cell(1,1);
6 b{1} = Set;
7 k = 100;
8 signatures = getSignatures(b, length(b), k);

```

É efetuada uma comparação de assinaturas para obter a similaridade de Jaccard, seguido de uma ordenação e limite de output.

```

1 size_genre = size(signatures_genre, 1);
2 similarities = zeros(1, size_genre);
3
4 for n = 1:size_genre
5     similarities(n) = 1 - (sum(signatures ~= signatures_genre(n, :)
6                               ) / k);
7 end

```

```

8 [sortedSimilarities, indices] = sort(similarities, 'descend');
9 topSimilarities = sortedSimilarities(1:5);
10 topTitles = movies(indices(1:5), 1);

```

Por fim, é realizado um print dos vetores, que contêm os demais títulos e similaridades.

```

1 fprintf("Top 5 Similar Titles:\n");
2 for i = 1:5
3     fprintf("\t%s - %f\n", topTitles{i}, topSimilarities(i));
4 end

```

O output é, como esperado:

```

1 Select one or more genres (separated by ','):
2     Comedy,Action,IMAX
3
4 Top 5 Similar Titles:
5     Night at the Museum: Battle of the Smithsonian - 1.000000
6     Men in Black III (M.III.B.) (M.I.B.) - 0.770000
7     I Love Trouble - 0.750000
8     Naked Gun 33 1/3: The Final Insult - 0.750000
9     Low Down Dirty Shame, A - 0.750000

```