universidade de aveiro
deti departamento de eletrónica,
telecomunicações e informática

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

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*.

```
function aux = muxDJB31MA(chave, seed, k)
len = length(chave);
chave = double(chave);
h = seed;
aux = zeros(1, k);
for i=1:len
    h = mod(31 * h + chave(i), 2^32 -1) ;
end
for i = 1:k
    h = mod(31 * h + i, 2^32 -1) ;
    aux(i) = h;
end
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*.

```
genres = strings([]);
[x, y] = size(movies);

for i = 1:x
    for j = 3:y
        if ~ismissing(movies{i, j})
            g = movies{i,j};
            if ~strcmp(g, '(no genres listed)') && ~ismember(
               convertCharsToStrings(movies{i, j}), genres)
                genres = [genres convertCharsToStrings(movies{i, j
                   })];
            end
        end
    end
end
```

3.3 Opção 2

3.3.1 Criação do filtro de Bloom

Para inicializar um filtro de bloom, recorre-se a um vetor com o tamanho passo como argumento da função `bloomFilterInitialization`.

Recorremos à formula $\frac{-n \times log(n)}{log(2)^2}$ para obter o tamanho adequado do filtro, onde n é a probabilidade de falsos positivos desejada. Escolhemos n como 0.001 porque consideramos ser um equilibro entre uma percentagem baixa de erro e o consumo de memória do filtro.

```
m = ceil(-n*log(0.001)/(log(2))^2);
genre_bloom = bloomFilterInitialization(m);

function bloom = bloomFilterInitialization(n)
bloom = zeros(1, n, 'uint16');
end
```

3.3.2 Inserção de elementos

Para inserir elementos, utilizamos a função `bloomFilterInsert`, onde se obtém o conjunto de *hashes* com base na *key* inserida, e, ao valor, se calcula o resto da divisão pelo tamanho do filtro de Bloom. Uma vez que este filtro envolve contagem, é feita a soma de 1 ao invés de definir como *true*.

Utilizámos esta função para inserir todos os géneros que se encontram no cell array criado pelo ficheiro CSV, ignorando os valores *missing*.

```
n = length(genres);
m = ceil(-n*log(0.001)/(log(2))^2);
k_g = round((m/n)*log(2));

for i = 1:x
    for j = 3:y
        if ~ismissing(movies{i, j})
            genre_bloom = bloomFilterInsert(genre_bloom, movies{i,
                j}, k_g);
        end
    end
end

function bloom = bloomFilterInsert(bloom, key, k)
m = length(bloom);
aux = muxDJB31MA(key, 127, k);
for i = 1:k
    hash = mod(aux(i), m) + 1;
    bloom(hash) = bloom(hash) + 1;
end
end
```

3.3.3 Pesquisa

Ao *char array* obtido através do *input*, é feita uma conversão para *string*, onde depois se invoca a função `bloomFilterCheck`.

Esta possui a mesma lógica da função `bloomFilterInsert` (ver secção 3.3.2), embora guarde os vários valores obtidos pela função de *hash* e retorne o menor.

Caso o valor obtido seja 0, significa que o género não existe.

```
genre = convertStringsToChars(input("Select a genre: ", "s"));
if ~ismember(genre, genres)
```

```
    fprintf("Genre not found!\n")
else
    count = bloomFilterCheck(genre_bloom, genre, k);
    fprintf("%d movies with genre %s\n", count, genre);
end

function count = bloomFilterCheck(bloom, key, k)
    m = length(bloom);
    aux = muxDJB31MA(key, 127, k);
    count = [];
    for i = 1:k
        key = [key num2str(i)];
        hash = mod(aux(i), m) + 1;
        count = [count bloom(hash)];
    end

    if ~isempty(count)
        count = min(count);
    else
        count = 0;
    end
end
```

O output desta opção é o seguinte:

```
Select a genre:
  Comedy

16124 movies with genre Comedy
```

3.4 Opção 3

3.4.1 Criação do filtro de Bloom

Para a criação do filtro, recorreu-se à mesma estratégia da opção 4 (ver secção 3.3.1).

3.4.2 Inserção de elementos

Para a inserção dos pares (Género, Ano) no bloom, recorreu-se à mesma função da secção 3.3.2, a `bloomFilterInsert`.

No entanto, uma vez que esta função aceita apenas strings, houve a necessidade de concatenar o par. Assim, para cada par, unimos ambos os valores numa string de formato "GéneroAno", e inserimos num vetor `genres_years`.

```
totalElements = x * (y - 2);
genres_years = strings(1, totalElements);
count = 1;

for i = 1:x
    for j = 3:y
        if ~ismissing(movies{i, j})
            genre_year = strcat(movies{i, j}, string(movies{i, 2}))
                ;
```

```
            genres_years(count) = genre_year;
            count = count + 1;
        end
    end
end

genres_years = genres_years(1:count-1);

n_gy = length(unique(genres_years));
m_gy = ceil(-n_gy*log(0.0001)/(log(2))^2);
k_gy = round((m_gy/n_gy)*log(2));

for i = 1:length(genres_years)
    genre_year_bloom = bloomFilterInsert(genre_year_bloom,
       convertStringsToChars(genres_years(i)), k_gy);
end
```

3.4.3 Pesquisa

Em primeiro lugar, é obtido o género e o ano através da função `strsplit`, que separa o input recebido.

Caso algum destes não se encontre no formato pedido, ou caso o género não exista, a opção termina com uma mensagem de erro.

```
a = input("Select a genre and a year (separated by ','): ", "s");
a = strsplit(a, ',');

genre = a{1};
if ~ismember(genre, genres)
    fprintf("Genre not found!\n")
    continue
end

year = a{2};
if isempty(str2num(year))
    fprintf("Invalid number!\n")
    continue
end
```

Por fim, é aplicada a mesma estratégia da opção 2, utilizando a função `bloomFilterCheck`. No entanto, esta é chamada duas vezes, uma para o género e outra para o par (género, ano), de modo a poder cruzar informações no output.

```
genre_year = convertStringsToChars(strcat(genre, year));
count_genre = bloomFilterCheck(genre_bloom, genre, k_g);
count_genre_year = bloomFilterCheck(genre_year_bloom, genre_year,
   k_gy);
fprintf("%d movies released in %s with genre %s\n",
   count_genre_year, year, genre);
```

O output é o seguinte:

```
Select a genre and a year (separated by ','):
```

```
    Comedy,1995

163 movies released in 1995 with genre Comedy
```

3.5 Opção 4

3.5.1 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.

```
[Set_title, Nm] = createMovieTitleStructure(movies);

function [Set, Nm] = createMovieTitleStructure(movies)
% For each movie, get its title split in shingles.
Nm = length(movies);
Set = cell(Nm,1);

for n = 1:Nm % Lines (movies)
    title = movies{n,1};
    for i = 1:length(title)-1
        Set{n} = [Set{n} convertCharsToStrings(title(i:i+1))];
    end
end
end
```

3.5.2 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.

```
signatures_title = getSignatures(Set_title, Nm, 100);

function signatures = getSignatures(Set, Nm, K)
signatures = inf(Nm, K);

for n = 1:Nm
    set_n = Set{n};
    for i = 1:length(set_n)
        key = num2str(set_n(i));
        h_out = muxDJB31MA(key, 127, K);
        signatures(n,:) = min(h_out, signatures(n,:));
    end
end
end
```

3.5.3 Pesquisa

Para a string introduzida na consola do MATLAB, são repetidos os passos das secções 3.5.1 e 3.5.2, de modo a obter as suas assinaturas.

```
a = input("Insert a string: ", "s");
Set = cell(1,1);

for i = 1:length(a)-1
    Set{1} = [Set{1} convertCharsToStrings(a(i:i+1))];
end

k = size(signatures_title, 2);
signatures = getSignatures(Set, 1, k);
```

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

```
size_titles = size(signatures_title, 1);
similarities = zeros(1, size_titles);


for n = 1:size_titles
    similarities(n) = 1 - (sum(signatures ~= signatures_title(n, :)
       ) / k);
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.

```
[sortedSimilarities, indices] = sort(similarities, 'descend');
topSimilarities = sortedSimilarities(1:5);
topTitles = movies(indices(1:5));
topGenres = Set_genre(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.

```
fprintf("Top 5 Similar Titles:\n");
concatenatedTitle = strings(length(topTitles{i}) + 1);
for i = 1:5
    displayString = sprintf("\t%s - %f - Genres: ", topTitles{i},
       topSimilarities(i));
    for j = 1:length(topGenres{i})
        displayString = strcat(displayString, convertCharsToStrings
           (topGenres{i}{j}), ", ");
    end

    fprintf('%s\n', displayString);
end
```

O output é, conforme esperado:

```
Insert a string:
  Love

```

```
Top 5 Similar Titles:
  Love - 1.000000
  Lover - 0.690000
  Love65 - 0.530000
  Loveling - 0.520000
  Lovelife - 0.520000
```

3.6 Opção 5

3.6.1 Estrutura de dados

A estrutura de dados é semelhante à encontrada para a opção 4 (ver secção 3.5.2). No entanto, ao invés de criar *shingles*, o set inclui os conjuntos de géneros para cada filme.

```
[Set_genre, Nm] = createMovieGenreStructure(movies);

function [Set, Nm] = createMovieGenreStructure(movies)
% For each movie, get its genres
Nm = length(movies);
Set = cell(Nm,1);

for n = 1:Nm % Lines (movies)
    for g = 3:12 % Columns (genres)
        if ~ismissing(movies{n, g})
            Set{n} = [Set{n} convertCharsToStrings(movies{n,g})];
        end
    end
end
end
```

3.6.2 Assinaturas

O conjunto de assinaturas para esta opção foi obtido da mesma forma que a opção 4, utilizando a *hash function* (ver secção 3.5.2).

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

3.6.3 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. Com isto, é possível verificar se algum dos géneros introduzidos não pertence ao conjunto de géneros existentes na matriz.

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

arevalid = true;
for i = 1:length(a)
    if ~ismember(a{i}, genres)
        fprintf("Genre %s is invalid!\n", a{i});
        arevalid = false;
        break
```

```
    end
end
if ~arevalid
    continue
end
```

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

```
Set = strings(length(a),1);
for i= 1:length(a)
    Set(i,1) = convertCharsToStrings(a(i));
end
b = cell(1,1);
b{1} = Set;
k = size(signatures_genre, 2);
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.

```
size_genre = size(signatures_genre, 1);
similarities = zeros(1, size_genre);

for n = 1:size_genre
    similarities(n) = 1 - (sum(signatures ~= signatures_genre(n, :)
       ) / k);
end

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

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

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

O output é, como esperado:

```
Select one or more genres (separated by ','):
  Comedy,Action,IMAX

Top 5 Similar Titles:
  Night at the Museum: Battle of the Smithsonian - 1.000000
  Men in Black III (M.III.B.) (M.I.B.) - 0.770000
  I Love Trouble - 0.750000
  Naked Gun 33 1/3: The Final Insult - 0.750000
  Low Down Dirty Shame, A - 0.750000
```