MPEI PL04

Universidade de Aveiro

José Miguel Silva, João Pedro Vieira

1.0

MPEI PL04

Departamento Eletrónica,Telecomunicações e Informática
Universidade de Aveiro

José Miguel Silva, João Pedro Vieira
(103248) jm.silva@ua.pt, (50458) joaopvieira@ua.pt

21/12/2023

Introdução

Este trabalho consistiu no desenvolvimento de uma aplicação em **MATLAB** para simular um sistema de pesquisa e disponibilização de filmes ao utilizador, tendo sido fornecido um ficheiros de dados, *"movies.csv"*, cujo conteudo detem uma lista filmes, indicando título, ano de estreia e géneros cinematográficos dos mesmos. Como requisito, foi necessário a produção e implementação de dois **MATLAB** *scripts*: um **script de apoio** para execução única onde é efetuada a leitura do ficheiro fornecido e criar todas as estruturas necessárias para a execução da aplicação (ficheiro: *"datamaker.m"* e outro para correr a aplicação e chamar todas as funções necessárias (ficheiro *"PL04*$_5$*0458*$_1$*03248"*).

Capítulo 1

Script de Apoio: datamaker.m

Este foi requisitado com o propósito de melhorar e otimizar o código, sendo executado prioritariamente e separadamente dos restantes ficheiros devido à sua funcionalidade de alta complexidade, o que tornaria lento o uso da aplicação.

Assim, este efetua a leitura do ficheiro de dados fornecido ("movies.csv", para gerar uma estrutura de dados complexa mas compreensível pelo MatLab, que comtem todos os dados necessários à implementação da aplicação e suas funcionalidades, nomeadamente:

- *1.1 Cria um conjunto de todos os géneros cinematográficos existentes no ficheiro de dados para a opção 1;*
- *1.2 Cria um Counting Bloom Filter de suporte à opção 2;*
- *1.2 Cria um Counting Bloom Filter de suporte à opção 3;*
- *1.3 Cria uma matriz de assinaturas de suporte à opção 4;*
- *1.3 Cria uma matriz de assinaturas de suporte à opção 5.*

```
%% [ DATAMAKER ]
%{
    UTILIDADE: Ler o ficheiro de entrada e criar a estrutura de dados
necessária que irá armazenar todos os dados necessários para implementar as
opções da aplicação:

%}
%% ########################################################################
%% Constroi estrutura de dados para armazenar a lista de géneros (únicos) por filme
    movieData = readcell('movies.csv','Delimiter',','); % Cell array dos filmes
    totalGenres = movieData(:, 3:end); % Cell array de géneros
    totalGenres = reshape(totalGenres,1,numel(totalGenres));
    temp = 1; allMovieGenders = {};

    % Remover "missing" cells
    for i = 1:length(totalGenres) -1
        if ismissing(totalGenres{i}) ~= 1
            allMovieGenders{temp} = totalGenres{i};
            temp = temp +1;
        end
    end

    % Obter géneros únicos
    genreList = unique(allMovieGenders);
    clear i; clear totalGenres; clear temp;
```

Figura 1.1: datamaker: Construção de estrutura de dados para lista de todos os géneros (únicos) disponíveis em movies.csv

```
%% ##########################################################################
%% Constroi a estrutura de dados para OPÇÃO 2

    % Parametros do Counting Bloom Filter
    m = length(allMovieGenders);
    n = 8 * m;
    %k = round((n * log(2)) / m);

    % Inicializar o Counting Bloom Filter
    BloomFilterGender = zeros(1, n);

    % Preencher o Counting Bloom Filter
    for i = 1:m
        hashcode = mod(hashstring(allMovieGenders{i}, n), n) + 1;
        BloomFilterGender(hashcode) = 1;
    end

%% ##########################################################################
%% Constroi a estrutura de dados para OPÇÃO 3

    totalYearsAndGenres = movieData(:, [2, 3:end]);
    totalYearsAndGenres = reshape(totalYearsAndGenres,1,numel(totalYearsAndGenres));
    temp = 1; allYearsGenders = {};

    % Remover "missing" cells
    for i = 1:length(totalYearsAndGenres) -1
        if ismissing(totalYearsAndGenres{i}) ~= 1
            allYearsGenders{temp} = totalYearsAndGenres{i};
            temp = temp +1;
        end
    end

     % Parametros do Counting Bloom Filter
    m = length(allYearsGenders);
    n = 8 * m;
    %k = round((n * log(2)) / m);

    % Inicializar o Counting Bloom Filter
    BloomFilterGenderYear = zeros(1, n);

    % Preencher o Counting Bloom Filter
    for i = 1:m
        hashcode = mod(hashstring(allYearsGenders{i}, n), n) + 1;
        BloomFilterGenderYear(hashcode) = 1;
    end
```

Figura 1.2: datamaker: Construção de estrutura de dados para as Opções 2 e 3 (Counting Bloom Filters)

```
%% ########################################################################
%% Constroi a estrutura de dados para OPÇÃO 4

    % Extract movie names and genres
    movieNames = movieData(:, 1);
    genres = movieData(:, 3:end);

    % Create a set of unique words across all movie names
    allWords = unique(strsplit(strjoin(movieNames, ' ')));

    % Parameteros do MinHash
    Nu = length(movieNames);
    K = 100;

    signaturesOp4 = inf(Nu, K);

    % Gerar assinaturas para cada filme
    for n = 1:Nu
        movieNameSet = unique(strsplit(movieNames{n}, ' '));
        for i = 1:length(movieNameSet)
            word = movieNameSet{i};
            h_out = muxDJB31MA(word, 127, K);
            signaturesOp4(n, :) = min(h_out, signaturesOp4(n, :));
        end
    end

%% ########################################################################
%% Constroi a estrutura de dados para OPÇÃO 5

    % Extract genres
    genres = movieData(:, 3:end);

    % Parameteros do MinHash
    Nu = size(genres, 1);
    K = 100;

    % Initialize signatures for Option 5
    signaturesOp5 = inf(Nu, K);

    % Generate signatures for each movie
    for n = 1:Nu
        genreSet = strsplit(genres{n}, ',');
        for i = 1:numel(genreSet)
            currentGenre = strtrim(genreSet{i});
            if ~isempty(currentGenre) && ~strcmp(currentGenre, 'N/A')
                h_out = muxDJB31MA(currentGenre, 127, K);
                signaturesOp5(n, :) = min(h_out, signaturesOp5(n, :));
            end
        end
    end



% Save necessary variables to a file
save('datamaker', 'genreList', 'movieData', 'BloomFilterGender', ...
    'BloomFilterGenderYear','signaturesOp4', 'signaturesOp5', 'm','K');
```

Figura 1.3: datamaker: Construção de estrutura de dados para as Opções 4 e 5 (minHash)

Capítulo 2

Script Principal

O ficheiro principal apresenta ao utilizador uma interface com todas as opções que o utilizador pode selecionar, sendo que a cada uma destas corresponderá a uma função.

As opções apresentadas ao utilizador serão:

2.1 1. Display available genres

Esta funcionalidade irá, através da leitura da estrutura de dados criada anteriormente, apresentar todos os géneros cinematográficos (únicos) dos filmes presentes no ficheiro movies.csv.

```
fprintf('The available genres are: ')
for i = 1: length(genreList)-2
    fprintf('%s, ',genreList{i})
end
fprintf('%s \n',genreList{length(genreList)-1})
```

Figura 2.1: Opção 1: Display available genres

2.2 2. Display Number of Movies by Gender

Esta funcionalidade irá através da utilização de um filtro de Bloom, iniciado no ficheiro de suporte, calcular de forma estimada o número de filmes existentes para um género introduzido pelo utilizador.

```
function moviesByGenre()
    load ('datamaker.mat','BloomFilterGender','genreList')
    estimatedCount = 0;
    genderSelect = input('Select a genre: ', 's');

    % Verificar se o genero existe
    if ~ismember(genderSelect, genreList)
        disp('Invalid genre. Please enter a valid genre: ');
        return;
    end

    % Calcula o numero estimado de filmes
    for i = 1:length(BloomFilterGender)
        index = mod(hashstring(genderSelect, length(BloomFilterGender)), length(BloomFilterGender)) + 1
        if BloomFilterGender(index) == 1
            estimatedCount = estimatedCount +1;
        end
    end

    fprintf('Estimated number of movies with genre %s: %d\n', genderSelect, estimatedCount);
end
```

Figura 2.2: Opção 2: Display Number of Movies by Gender

2.3 3. Display Number of Movies by Gender and Year

Esta funcionalidade irá através da utilização de um filtro de Bloom, iniciado no ficheiro de suporte, calcular de forma estimada o número de filmes existentes para um género e um ano especifico introduzido pelo utilizador.

```
function moviesByGenreAndYear()
    load('datamaker.mat', 'genreList', 'BloomFilterGenderYear', 'm');
    n = 8 * m;
    k = round((n * log(2)) / m);


    % Pede ao Utilizador o Input(Separado por ',')
    userInput = input('Select a genre and year (e.g., Action,2000): ', 's');

    % Separa o Input em Genero e Ano
    userInputParts = strsplit(userInput, ',');

    % Check para caso faltem argumetos
    if numel(userInputParts) ~= 2
        disp('Invalid input. Please provide both genre and year separated by a comma.');
        return;
    end

    genreSelect = userInputParts{1};
    yearSelect = str2double(userInputParts{2});

    % Check para caso o genero não exista
    if ~ismember(genreSelect, genreList)
        disp('Invalid genre. Please enter a valid genre.');
        return;
    end
```

Figura 2.3: Opção 3: Display Number of Movies by Gender and Year pt.1

```
    % Check para caso o ano não seja válido
    if isnan(yearSelect) || ~isreal(yearSelect) || yearSelect < 1
        disp('Invalid year. Please enter a valid positive integer.');
        return;
    end

    % Inicia o Counting Bloom Filter para o ano e genero
    genreIndex = find(strcmp(genreList, genreSelect));
    BloomFilterGenreYear = BloomFilterGenderYear(1,genreIndex);

    % Estima o numero de filmes
    estimatedCount = movieCount(BloomFilterGenreYear, m, k);
    fprintf('Estimated number of movies with genre %s in year %d: %d\n', genreSelect, yearSelect, est
end

function count = movieCount(BloomFilterGenreYear, m, k)
    estimate = 0;
    % Repete varias vezes o hash
    for i = 1:k
        h = DJB31MA([num2str(i) BloomFilterGenreYear], 127);
        h = mod(h, numel(BloomFilterGenreYear)) + 1;
        estimate = min(estimate, BloomFilterGenreYear(h));
    end
    % Aprefeiçoa a estimativa com a formula
    count = round(-m * log(1 - estimate / m) / k);

    % Verifica se a estimativa é positiva
    count = max(0, count);
end
```

Figura 2.4: Opção 3: Display Number of Movies by Gender and Year pt.2

2.4 4. Search Movies by Title

Esta funcionalidade consiste em realizar pesquisa de um filme, pelo seu título, apresentando os 5 filmes cujo título se aproxime mais ao inserido pelo utilizador.

Esta pesquisa é realizada através do cálculo da distância de Jaccard, utilizando o método minHash, entre o título inserido e todos os filmes guardados, apresentando também o valor da aproximação.

```
load('datamaker.mat', 'movieData', 'signaturesOp4');

% Pede o nome do filme
userInput = input('Insert a string: ', 's');

% Cria as Assinaturas
userSignature = generateSignature(userInput);

% Calcula a Semelhança entre o Nome posto pelo User e Os Filmes
similarities = calculateSimilarities(userSignature, signaturesOp4);

% Ordena Por Ordem Decrescente de Jaccard
[~, sortedIndices] = sort(similarities, 'descend');

% Diplay dos 5 Filmes Mais Proximos
disp('Top 5 Movies Similar to the Inserted String:');
disp('-----------------------------------------');
for i = 1:min(5, length(sortedIndices))
    movieIndex = sortedIndices(i);
    movieName = movieData{movieIndex, 1};
    genre = movieData(movieIndex, 3:end);
    similarity = similarities(movieIndex);

    %Nº na ordem e Titulo
    fprintf('\n %d. %s (Genres: ', i, movieName);

    % Genero do Filme
    for k = 1:length(genre)-1
        if ismissing(genre{k}) == 0
            fprintf('%s; ', genre{k})
```

Figura 2.5: Opção 4: Search Movies by Title pt.1

```
                fprintf('%s; ', genre{k})
            end
        end

        %Jaccard Similarity
        fprintf(')(Jaccard Similarity: %.4f)', similarity);
        % ------------------------------------------------
    end
end


function signature = generateSignature(inputString)
    K = 100;

    % Converte String para um Set de Words
    words = unique(strsplit(inputString, ' '));

    % Cria MinHash para a String
    signature = inf(1, K);
    for i = 1:length(words)
        word = words{i};
        h_out = muxDJB31MA(word, 127, K);
        signature = min(h_out, signature);
    end
end

function similarities = calculateSimilarities(userSignature, movieSignatures)
    similarities = sum(userSignature == movieSignatures, 2) / length(userSignature);
end
```

Figura 2.6: Opção 4: Search Movies by Title pt.2

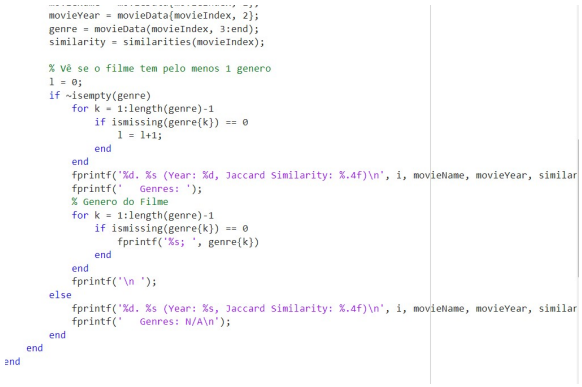
2.5 5. Search Movies by Genres

Esta funcionalidade possibilita o utilizador a pesquisar filmes por um ou mais géneros, introduzindo os géneros pretendidos esta funcionalidade irá calcular os 5 filmes cujos géneros se aproximem mais aos introduzidos pelo utilizador.

Tal como a funcionalidade anterior, o calculo também é feito através da distância de Jaccard, utilizando o minHash, entre os géneros de cada filme e o os introduzidos pelo utilizador.

```
function minHashOp5()
    load('datamaker.mat', 'movieData', 'signaturesOp5', 'genreList');

    userInput = input('Select one or more genres (separated by '','' ): ', 's');
    selectedGenres = strsplit(userInput, ',');

    % Verificar se os gêneros selecionados são válidos
    if ~all(ismember(selectedGenres, genreList))
        disp('Invalid genres. Please enter valid genres.');
        return;
    end

    % Gerar assinatura para os gêneros selecionados
    userSignature = generateSignature(selectedGenres);

    % Calcular a semelhança de Jaccard entre input e os filmes
    similarities = calculateSimilarities(userSignature, signaturesOp5);

    % Ordenar filmes com base na semelhança (ordem decrescente)
    [~, sortedIndices] = sort(similarities, 'descend');

    % Exiba os 5 melhores filmes (semelhança Jaccard)
    disp('Top 5 movies similar to selected genres:');
    disp('----------------------------------------');
    for i = 1:min(5, length(sortedIndices))
        movieIndex = sortedIndices(i);
        movieName = movieData{movieIndex, 1};
        movieYear = movieData{movieIndex, 2};
        genre = movieData(movieIndex, 3:end);
```

Figura 2.7: Opção 5: Search Movies by Genres pt.1



```
        movieYear = movieData(movieIndex, 2);
        genre = movieData(movieIndex, 3:end);
        similarity = similarities(movieIndex);

        % Vê se o filme tem pelo menos 1 genero
        l = 0;
        if ~isempty(genre)
            for k = 1:length(genre)-1
                if ismissing(genre{k}) == 0
                    l = l+1;
                end
            end
            fprintf('%d. %s (Year: %d, Jaccard Similarity: %.4f)\n', i, movieName, movieYear, similar
            fprintf('   Genres: ');
            % Genero do Filme
            for k = 1:length(genre)-1
                if ismissing(genre{k}) == 0
                    fprintf('%s; ', genre{k})
                end
            end
            fprintf('\n ');
        else
            fprintf('%d. %s (Year: %s, Jaccard Similarity: %.4f)\n', i, movieName, movieYear, similar
            fprintf('   Genres: N/A\n');
        end
    end
end
```

Figura 2.8: Opção 5: Search Movies by Genres pt.2

```
function signature = generateSignature(selectedGenres)
    K = 100;

    % Gera MinHash para os generos
    signature = inf(1, K);
    for i = 1:length(selectedGenres)
        genre = selectedGenres{i};
        if ~ismissing(genre)
            h_out = muxDJB31MA(genre, 127, K);
            signature = min(h_out, signature);
        end
    end
end

function similarities = calculateSimilarities(userSignature, movieSignatures)
    intersection = sum(userSignature == movieSignatures, 2);
    unionSet = sum(userSignature | movieSignatures, 2);
    similarities = intersection ./ unionSet;
end
```

Figura 2.9: Opção 5: Search Movies by Genres pt.3

2.6 6. Exit

Esta funcionalidade permite ao utilizador terminar a execução do programa.

Capítulo 3

Option 2: moviesByGenre()

Na opção 2, é chamada a função moviesByGenre() a qual recorre à informação (BloomFilterGender e genreList) gerada pelo datamaker.m e implementa um Counting Bloom Filter, que irá contar o número (estimado) de filmes cujo género corresponde ao selecionado, tal como podemos verificar nas figuras seguintes:

```
function moviesByGenre()
    load ('datamaker.mat','BloomFilterGender','genreList')
    estimatedCount = 0;
    genderSelect = input('Select a genre: ', 's');

    % Verificar se o genero existe
    if ~ismember(genderSelect, genreList)
        disp('Invalid genre. Please enter a valid genre: ');
        return;
    end

    for i = 1:length(BloomFilterGender)
        index = mod(hashstring(genderSelect, length(BloomFilterGender)), length(BloomFilterGender)) + 1;
        if BloomFilterGender(index) == 1
            estimatedCount = estimatedCount +1;
        end
    end

    fprintf('Estimated number of movies with genre %s: %d\n', genderSelect, estimatedCount);
end
```

Figura 3.1: Opção 2: moviesByGenre - Counting Bloom Filter

```
 SELECTION MENU:

1 - Display available genres
2 - Number of movies of a genre
3 - Number of movies of a genre on a given year
4 - Search movie titles
5 - Serch movies based on genres
6 - Exit

Select an option: 2
Select a genre: Action
Estimated number of movies with genre Action: 834992
```

Figura 3.2: Opção 2: Resultado de uma pesquisa por género - Action

Capítulo 4

Option 3: moviesByGenreAndYear()

Na opção 2, é chamada a função moviesByGenreAndYear() a qual recorre à informação gerada pelo datamaker.m e implementa um Counting Bloom Filter, que irá contar o número (estimado) de filmes cujo género e ano correspondem aos selecionados.

Infelizmente, não nos foi possível colocar esta opção totalmente funcional, dando constantemente uma contagem de 0 para quaisquer valores válidos de input (genero,year).

```
 SELECTION MENU:

1 - Display available genres
2 - Number of movies of a genre
3 - Number of movies of a genre on a given year
4 - Search movie titles
5 - Serch movies based on genres
6 - Exit

Select an option: 2
Select a genre: Action
Estimated number of movies with genre Action: 834992
```

Figura 4.1: Opção 2: Resultado de uma pesquisa por género - Action

Capítulo 5

Opções 4 e 5: minHash()

Capítulo 6

Conclusão

Em suma,