Métodos Probabilísticos de Engenharia Informática

Turma P8

Ano Letivo 2022/2023

Patrícia Cardoso, Inês Santos

Métodos Probabilísticos de Engenharia Informática

Turma P8

(103243) Patrícia Cardoso
patriciarcardoso@ua.pt

(103477) Inês Santos
ines.santos20@ua.pt

7 de janeiro de 2023

Índice

Capítulo 1

Introdução

Neste guião, foi-nos proposto que se desenvolvesse, em Matlab, algumas funcionalidades de um sistema (aplicação) online de disponibilização de filmes.

Nesse sistema, cada utilizador está identificado com um ID, bem como cada filme existente na lista (sendo ambos os IDs números inteiros e positivos). Cada utilizador vê determinado filme, e consequentemente, atribui-lhe uma dada avaliação (de 1 a 5).

Como ponto de partida, precisámos de 2 documentos que nos foram fornecidos:

- **u.data.txt :** que possui na sua 1ª coluna os IDs dos utilizadores, na sua 2ª coluna os IDs dos filmes vistos por cada utilizador, e na 3ª a avaliação atribuída a esse filme pelo utilizador correspondente.
- **u_item.txt :** gerado a partir do ficheiro u.data.txt, contém 20 colunas e está organizado por tabs. Cada linha n corresponde ao filme cujo ID n se encontra na 2ª coluna do ficheiro u.data. A 1º coluna contém o nome do filme, e as restantes (2 a 20 - que correspondem aos diferentes géneros que os filmes podem tomar) estão a 1 ou a 0 consoante os géneros em que esse filme é classificado (se a coluna estiver com o valor 1, então o filme dessa linha é classificado como segundo esse ou outros géneros. Se estiver a 0, não pertence ao género de filmes correspondentes a essa coluna).

A aplicação proposta funciona tendo por base um menu, em que, inicialmente, se pede ao utilizador atual que insira o seu ID (entre 1 e 943), havendo uma validação do mesmo. Posteriormente, o utilizador terá a oportunidade de escolher entre uma de 5 opções, consoante a funcionalidade que deseja ver implementada.

O código Matlab em seguida demonstra como foi feito esse menu da aplicação.

```
    prompt = "Insert User ID (1 to 943): ";
id = input(prompt);
if id >= 1 && id <= 943
    disp("1 - Your movies");
    disp("2 - Suggestion of movies based on other users");
    disp("3 - Suggestion of movies based on already evaluated movies");
    disp("4 - Search Title");
    disp("5 - Exit");
    prompt_option = "Select choice:";
    option = input(prompt_option);
    switch option
        case 1
            yourMovies(id);
        case 2
            moviesBasedOtherUsers(id);
        case 3
            disp('3');
        case 4
            disp('4');
        case 5
            quit(1);
    end
end
```

Capítulo 2

Opção 1

De forma a listar os títulos dos filmes que o utilizador atual (ID inserido e validado previamente) viu, criou-se o cell aray *dic{i,j}* que contém a informação da linha **i** e da coluna **j** do ficheiro u_item.txt . A procura é feita usando o cell array, comparando o ID inserido com os IDs da 1ª coluna do ficheiro u.data, e depois de se encontrar o ID do utilizador, faz-se uma procura pela 2ª coluna de u.data de modo a obterem-se os IDs dos filmes vistos pelo utilizador.

De forma a descobrir o nº de vezes que cada filme visto pelo utilizador atual foi avaliado por todos os utilizadores, usámos um *Counting Bloom Filter*, que permite, ao contrário dos Bloom Filters simples, obter informação sobre a multiplicidade de um elemento ou mais elementos de um conjunto C.

Foi criado um Counting Bloom Filter para cada filme visto pelo utilizador atual através de uma hash function para cada filme, e acrescentavam-se elementos ao filtro (função incrementar()), ou seja, havia um contador por utilizador,para cada filme avaliado. De seguida, para testar a multiplicidade, usou-se a função contagem(), na qual se obteve a estimativa com menos erro ao se calcular o mínimo armazenado nesses contadores para o mesmo filme. E repetiu-se este processo para cada filme visto pelo utilizador atual.

```
function [] = yourMovies(user_id)
    udata = load('u.data.txt'); %Load data from u.data
    dic = readcell('u_item.txt','Delimiter','\t'); %Read data from u_item
    saw = udata(udata(:,1) == user_id,2); %Compara o user_id introduzido com cada id
                        % do utilizador presente na primeira coluna do ficheiro u_data.
                                            %Depois de encontrar esse id do
                                            %utilizador procura na segunda coluna
                                            %do u_data os ids dos filmes vistos
                                            %pelo utilizador

    titles = {}; %cell array that contains the titles of the movies
    for i = 1:length(saw)
```

```
        movie_id = saw(i); %saw contém os ids dos movie vistos por aquele utilizador
        movie_title = dic{movie_id,1};%movie title for the movie with the given id
        titles{end+1} = movie_title; % assure that the movie title is appended to
                                     % the list of the titles in the correct order
    end

    fprintf("Títulos dos filmes vistos pelo utilizador atual:\n");
    for i=1:length(titles)
        fprintf("%d. %s\n",i,titles{i});
    end
    %disp(length(titles));

    %Racicionio counting bloom filter -> Numero de vezes que um filme foi
    %avaliado por todos os utilizadores
    %->Para cada filme aplicamos uma função de dispersão que resultará num
    %hashcode, incrementamos em um essa posição no bloom filter
    %->mas como sabemos qual foi de facto o numero de vezes? o valor minimo
    %entre os varios contadores correspondentes ao elemento dá-nos uma
    %estimativa desse numero

    bf = inicializar(5046); %porque sao 1682 filmes
    for i = 1:size(udata,1) %para todos os users
        movie_id = udata(i,2); %ids_filmes
        bf = incrementar(bf,movie_id,3); %incrementar o contador do id
                                         %do filme no filtro de bloom
    end
    movie_id = saw(1); %id do filme que eu quero analisar

    str = '----------------------------------------------------------------------';
    fprintf("%s\n",str);
    for i=1:length(titles)
        movie_id = saw(i);
        rate_count = contagem(bf,movie_id,3,5046); %determinar a estimativa
        fprintf("O filme %s foi avaliado %d vezes por todos os utilizadores.
                                                    \n",titles{i},rate_count);
    end
end
```

Função inicializar() :

```
    function b = inicializar(n) %n é o tamanho do vetor,do bloom filter
    b = zeros(1,n); %inicializa o vetor a 0s
end
```

Função incrementar() :

```
    function b = incrementar(b,elem,k) %b ->vetor,elem->elemento que pretendemos
    adicionar, k->número de funções de dispersão
    n = length(b); %tamanho do vector
    for i=1:k
        h = DJB31MA(elem,127 + i); %h é o hashcode,vai ser diferente por causa do hashcode
        h = mod(h,n) + 1; %valor entre 1 e n
        b(h) = b(h) + 1; %incrementa a posição em 1
        %pretendo mostrar como o vetor ficou por isso dou "return" de b
    end
end
```

Hash Function usada neste guião:

```
    function h= DJB31MA( chave, seed)
% implementação da hash function DJB31MA com base no algoritmo obtido
% no resumo 2014(PJF) que está em C
%
%  chave    array de caracteres com a chave
%  seed     semente que permite obter vários hash codes para a mesma chave
%
%  h        hashcode devolvido
len= length(chave);
chave= double(chave);
h= seed;
for i=1:len
    h = mod(31 * h + chave(i), 2^32 -1) ;
end
```

Função contagem() :

```
    function minimum = contagem(b,elem,k,n) %n->length of the array
    minimum = Inf;
    for i=1:k
        h = DJB31MA(elem,127 + i); %different seed to guarantee that the hash code
                                   % will be different
        h = mod(h,n) + 1;
        minimum = min(minimum,b(h)); %colocar o minimo valor
    end
end
```

Capítulo 3

Opção 2

De modo a se determinarem os 3 utilizadores mas similares ao utilizador atual, a função carrega os dados do ficheiro u.data.txt e conta o número de classificações com uma pontuação de 3 ou superior usando um Counting Bloom Filter. Depois, encontra os IDs de utilizadores únicos e os IDs de filmes correspondentes classificados por cada utilizador e cria uma matriz de assinaturas com os vetores minhash para os conjuntos de IDs de filmes usando a função minhash2.
A função determina os três utilizadores que são mais semelhantes ao utilizador atual (com base nos conjuntos de filmes classificados com uma pontuação de 3 ou superior) e imprime os seus IDs.
A função encontra os títulos de filmes que foram classificados por pelo menos um dos três utilizadores mais semelhantes e não foram classificados pelo utilizador atual. Para isso, lê os dados do ficheiro u_item.txt e obtém os IDs de filmes classificados pelo utilizador atual. Em seguida, percorre os IDs de filmes classificados pelos três utilizadores mais semelhantes e verifica se cada um desses filmes foi classificado pelo utilizador atual. Se um filme não foi classificado pelo utilizador atual, a função imprime o título do filme.

Função que executa a 2ª opção do menu da aplicação:

```
function [] = moviesBasedOtherUsers(user_id)
u_data = load('u.data.txt'); %Load data from u.data
%Couting bloom filter para armazenamento do número de avaliações com
%nota superior ou igual a 3
bf = inicializar(5046); %porque sao 1682 filmes
for i = 1:size(u_data,1) %para todos os users
    if u_data(1,3) >= 3 %se a nota é igual ou superior a 3
        movie_id = u_data(i,2); %ids_filmes
        bf = incrementar(bf,movie_id,3); %incrementar o contador
                    %do id do filme no filtro de bloom
    end
```

```
end


%Find the unique user ids and the corresponding movie ids rated by each
%user
[user_ids,~,subs] = unique(u_data(:,1)); %coloca em user_ids os valores
%da primeira coluna de u_data,~->ignores the second output of
%the unique function
movie_ids = cell(length(user_ids),1);
for i = 1:length(user_ids)
    movie_ids{i} = u_data(subs == i,2);
end
%Create a signature matriz with the MinHash vectors for the sets of
%movie ids
k = 100;
signature_matrix = zeros(k,length(user_ids)); %Initialize
for i = 1:length(user_ids)
    %Create a set of movies ids using the membro function
    movie_set = membro(bf,movie_ids{i},k);
    %Create a minhash vector using the minhash function
    minhash_vector = minhash2(movie_set,k);
    %store the minhash vector in the signature vector
    signature_matrix(:,i) = minhash_vector;
end

%Determinar os 3 utilizadores mais similares ao utilizador atual(em
%termos de conjuntos de filmes avaliados com nota superior ou igual a
%3)
%user_id = 10;
user_index = find(user_ids == user_id);
user_signature = signature_matrix(:,user_index);
sim_scores = zeros(length(user_ids),1);

for i = 1:length(user_ids)
    if i == user_index
        %Skip the current user
        continue;
    end
    otheruser_signature = signature_matrix(:,i);

    %Calculate the Jaccard similarity between the current user and this user
    common_elem = sum(user_signature == otheruser_signature);
    total_elem = sum(user_signature ~= otheruser_signature);
    sim_scores(i) = common_elem/total_elem;

    %Find the three users with the highest similarity
```

```
        [~,sort_indexs] = sort(sim_scores,'descend');
        most_sim_user_indexs = sort_indexs(1:3);
        most_sim_user_ids = user_ids(most_sim_user_indexs);
    end
    fprintf("Os três utilizadores mais semelhantes ao utilizador %d são:
             \n",user_id);
    for i=length(most_sim_user_ids)
        fprintf("%d\n",most_sim_user_ids);
    end


    % Apresenta os títulos dos filmes que foram avaliados por pelo menos um
    % dos 3 utilizadores e que ainda não foram avaliados pelo utilizador
    % atual
    dic = readcell('u_item.txt','Delimiter','\t'); %Read data from u_item
    %Get the movie IDs rated by the current user
    user_rated_movies_ids = movie_ids{user_index};

    titles = {}; %initialize

    for i = 1:3 %for each similar user
        %Get the movie ids rated by the current most similar user
        most_sim_user_rated_movie_ids = movie_ids{most_sim_user_indexs(i)};
        %Find the movie ids rated by the most similar user that have not
        %been rated by the current user
        n_movie_ids = setdiff(most_sim_user_rated_movie_ids,user_rated_movies_ids);
        %Find the titles of the movies with the new movie IDs

        for j = 1:length(n_movie_ids)
            movie_id = n_movie_ids(j);
            movie_title = dic{movie_id,1};
            titles{end+1} = movie_title;
        end
    end

    fprintf("Títulos dos filmes que foram avaliados por pelo menos um dos
        3 utilizadores e que ainda não foram avaliados pelo utilizador atual:\n");
    for i=1:length(titles)
        fprintf("%s\n",titles{i});
    end
end
```

Algoritmo MinHash usado:

```
    function minhash = minhash2(movie_set,k)
    %Initialize
    minhash = inf(k,1);
    %Generate k random permutations of the movie set permutations
    permutations = cell(k,1);
    for i=1:k
        permutations{i} = randperm(length(movie_set));
    end
    %For each permutation,find the first movie that is in the set and
    %update the corresponding element in the minhash
    for i=1:k
        for j = 1:length(permutations{i})
            if movie_set(permutations{i}(j))== 1
                minhash(i) = permutations{i}(j);
            break
            end
        end
    end
end
```

Função membro() :

```
    function cond = membro(b,elem,k) %b->bloom filter,elem->elemento,k->função de dispersão
    n = length(b);
    cond = true;
    for i=1:k
        h = DJB31MA(elem,127);
        h = mod(h,n) + 1; %valor entre 1 e n
        if b(h) <= 0 %requer que todos os elementos sejam não nulos
            cond = false;
            break;
        end
    end
end
```

Capítulo 4

Opção 3

Capítulo 5

Opção 4

Esta opção devolve os 5 nomes de filmes com títulos mais similares à string introduzida.O utilizador insere uma string com o nome de um filme(ou parte do nome). É usando o método minhash que permite calcular a similaridade entre a string e os títulos dos filmes.Através da similaridade de Jaccard esta operação pode ser realizada. Primeiro,é lido do ficheiro u_item.txt a lista de títulos de filmes e armazenados num cell array.Depois é inicializada a matriz de assinaturas com os vetores minhash associados aos títulos dos filmes.É calculada a similaridade de Jaccard entre o vetor minhash da string introduzida e os vetores minhash dos filmes. Os indíces dos cinco nomes de filmes mais similares à string introduzida são determinados em ordem decrescente e consideram-se os primeiros cinco indícies.

Função que executa a 4ª opção do menu da aplicação:

```
function sim_titles = searchTitle(string)
    dic = readcell('u_item.txt','Delimiter','\t'); %Read data from u_item
    titles = dic(:,1); %get the movie titles
    titles_number = size(titles,1);
    k = 100; %columns numver of signature matrix
    signature_matrix = zeros(titles_number,k); %initialize the matrix
                                                %to store the vectors
    %Creates a minHash vector for each movie title and add to the signature
    %matrix
    for i=1:titles_number
        minhash = minhash4(titles{i});
        minhash = horzcat(minhash,zeros(1,k-length(minhash)));
    %disp(length(minhash)); - não são do mesmo tamanho, é preciso ajustar o
    %tamanho do vetor usando por exemplo a função horzcat

    %Add the minhash vector to the signature matrix at line corresponding to
    %the title of the movie
```

```
        signature_matrix(i,:) = minhash;
    end

    minhash_search = minhash4(string); %calculan o vetor minhash para a string
    %calcula a sim de jaccard entre o vetor minHash da string de busca e os
    %vetor minhash dos filmes
    sim = zeros(size(signature_matrix,1),1);
    for i = 1:size(signature_matrix,1)
        sim(i)=jaccard(minhash_search,signature_matrix(i,:));
    end
    [~,indices] = sort(sim,'descend');
    five = indices(1:5);
    sim_titles = titles(five);
    for i=1:length(sim_titles)
        fprintf('%s\n',sim_titles{i});
    end
end
```

Função minhash() :

```
function minhash = minhash4(title)
   shingles_size = 3; %Qual será o melhor?
   shingles = createShingles(title,shingles_size);
   k = 100; %qual o melhor numero de k?

   hash_codes = zeros(k,length(shingles)); %Valor de k para cada shingle

   %Valores aleatórios para a função de hash
   a = randi([1,intmax('uint32')],k,1);
   b = randi([0,intmax('uint32')],k,1);

   %calcula os hash codes para cada shingle
   for i=1:length(shingles)
       for j=1:length(shingles)
           hash_codes(i,j) = mod((a(i)*DJB31MA(shingles{i},127+i) + b(i)),
                                        intmax('uint32'));
       end
   end
   minhash = min(hash_codes(1:k));
end
```

Função createShingles() :

```
function shingles = createShingles(title,shingleSize)
    title = strtrim(title); %Remove espaços em branco do título de filme

    %Divide o título de filme em shingles
    shingles = cell(1,ceil(length(title)/shingleSize));
    index = 1;
    for i=1:shingleSize:length(title)
        if i+shingleSize-1 <= length(title)
            shingles{index} = title(i:i+shingleSize-1);
        else
            shingles{index} = title(i:end);
        end
        index = index + 1;
    end
end
```

Função Jaccard()

```
function sim = jaccard(c1,c2)
    c1 = logical(c1);
    c2 = logical(c2);

    intersection = sum(c1 & c2);
    union = sum(c1 | c2);
    sim = intersection/union;
end
```