MPEI Teste 2

Universidade de Aveiro

Telmo Sauce (104428), Tiago Mostardinha (103944)

MPEI Teste 2

DETI

Universidade de Aveiro

Telmo Sauce (104428), Tiago Mostardinha (103944)

Ano letivo 2022/2023

Índice

Capítulo 1

Introdução

Foi nos proposto a resolução de um guião em âmbito da avaliação de Métodos Probabilísticos de Engenheria Informática, onde desenvolvemos dois Scripts em Matlab, *"app.m"* e *"dados.m"*. O primeiro corre uma única vez para criar as estruturas de dados associadas aos user e aos files e para ler dois ficheiros de entrada, *"u.data"* que contem na primeira coluna o ID dos users, na segunda o ID dos filmes avaliados por cada user e por fim na terceira coluna a avaliação atribuída por cada user, e o *"u_ item.txt"* onde se encontra uma lista de todos os filmes por ordem crescente do seu ID e uma lista de zeros e uns em que o seu índice corresponde ao género do filme. Para criação das estruturas de dados foi necessário o uso do Script *"DJB31MA.m"* fornecido nas aulas. O segundo Script lê todas as estruturas de dados previamente guardadas e implementa a aplicação com que o user irá interagir.

Capítulo 2

Leitura dos ficheiros

```
%% read data files

%reads u.data
load("u.data");

%reads Films.txt
listMoviesBin = readcell('u_item.txt', 'Delimiter','\t
    ');
movie_genre = ["" "unknown" "Action" "Adventure" "
    Animation" "Children's" "Comedy" "Crime" "
    Documentary" "Drama" "Fantasy" "Film-Noir" "Horror"
     "Musical" "Mystery" "Romance" "Sci-fi" "Thriller"
    "War" "Western"];

%generos de boolean para text
for i=1 : length(listMoviesBin)
    listMovies(i,1)=listMoviesBin(i,1);
    k=1;
    for j=2 : length(listMoviesBin(i,:))
        if isequal(listMoviesBin{i,j},1)
            k = k+1;
            listMovies(i,k) = {movie_genre(1,j)};
        end
    end
end

%all users
users = unique(u(:,1)); % Extrai os IDs dos
    utilizadores
Nu = length(users); % Numero de utilizadores
```

```
fprintf("Read Done!\n");
```

Explicação

Este trecho de código irá ler os dois files fornecidos e vai guardar o *"u.data"* em uma matriz,"u"e o *"u_ item.txt"* em um vetor, *"listMoviesBin"*. o vetor *"listMoviesBin"* passara por um loop para atribuir os géneros de cada um e ser guardado em *"listMovies"*

Capítulo 3

Opção 1

A aplicação irá listar os títulos dos filmes que o utilizador atual viu e para cada titulo o número de vezes que foi avaliado por todos os users.

3.1 dados.m

```
%% Encontra Filmes de todos os users
Set = cell(Nu,1);
for n = 1:Nu
    % Obtem os filmes de cada user
    ind = find(u(:,1) == users(n));
    Set{n} = [Set{n} u(ind,2) u(ind,3)];
end
fprintf("Filmes do Users Done!\n");

%% Counting bloom filter
for i=1:length(uData)
    vec{i,1} = u(i,2);
end

n =length(vec) * 8;
m = length(vec);
k = 5;

BF = init(n);

for i = 1:m
    BF = insert(vec{i}, BF, k);
end

fprintf("Counting Bloom Filter Done!\n");
```

```
%% funtions Bloom Filter

function BF = init(n)
    BF = zeros(1,n);
end

function BF = insert(elemento, BF, k)
    n = length(BF);
    for i = 1:k
        elemento = [elemento num2str(i)];
        h = DJB31MA(elemento, 127);
        h = mod(h,n) + 1; %para dar valor entre 1 e n
           para por no BF
        BF(h) = BF(h) + 1;
    end
end
```

Explicação

Neste Script foi criado uma variável *"Set"* que contem os IDs de todos os filmes avaliados por cada user e a sua respetiva avaliação. Para alem disso foi inicializada um Counting Bloom Filter que irá contar o número de vezes que cada filme foi avaliado pelos users, neste *"m"* será o número de filmes que iram ser contados, *"n"* é oito vezes maior, este número foi o que foi usado nas aulas não sendo um número muito elevado, para não ter um grande gasto na memoria, nem um número muito pequeno para evitar um grande número de falsos positivos. O *"k"* será calculado pela formula, (0,68 * m)/ n, logo k será igual a 5 arredondado.

3.2 app.m

```
%% yourMovies Function
function yourMovies(user, Set, listMovies, BF)
    k = 5;
    fprintf('\nFilmes vistos:\n');
    for i = 1:length(Set{user})
        fprintf('%s , ID:%d , CBF: %d\n', listMovies{
           Set{user}(i)}, Set{user}(i), couting(Set{
           user}(i), BF, k));
    end
    pause;
end
```

```
%Bloom Filter Funtion
function count = couting(elemento, BF, k)
    n = length(BF);
    count = 0;
    for i = 1:k
        elemento = [elemento num2str(i)];
        h = DJB31MA(elemento, 127);
        h = mod(h,n) + 1; %para dar valor entre 1 e n
           para por no BF
        count = BF(h);
    end
end
```

Explicação

Aqui a *"app.m"* usa a estrutura de dados *"Set"* e *"listMovies"* para listar os nomes dos filmes do user atual, para alem disso é usado a função counting para saber quantas vezes o filme foi avaliado, este número está armazenado em *"BF"*.

Capítulo 4

Opção 2

A aplicação nesta opção irá disponibilizar ao user os filmes, que foram avaliados com nota superior a 3 ou igual, dos 3 users mais similares ao user atual.

4.1 dados.m

```
%% usersByMovie

K = 200;  % Numero de funcoes de dispersao
usersByMovie = inf(Nu,K);
for i = 1:Nu
    conjunto = Set{i};
    for j = 1:length(conjunto)
        idx = find(u(:,1) == users(i) & u(:,2) ==
           conjunto(j));
        if( u(idx,3) >= 3)
            chave = char(conjunto(j));
            hash = zeros(1,K);
            for kk = 1:K
                chave = [chave num2str(kk)];
                hash(kk) = DJB31MA(chave,127);
            end
            usersByMovie(i,:) = min([usersByMovie(i,:)
               ; hash]);  % Valor minimo da hash para
               este titulo
        end
    end
end

fprintf("usersByMovie Done!\n");
```

Explicação

No script *dados.m* foi criado uma estrutura de dados *"usersByMovie"* onde será armazenada as *Hash de cada user* para masis tarde na *"app.m"* serem comparadas e determinado a sua distância de jaccard, para poder destinguir que users são mais similares. Os filmes com avaliação inferior a 3 serão descartados.

4.2 app.m

```
%% otherUserSugestions
function otherUserSugestions (Nu, usersByMovie, user,
   Set, listMovies)
    K = 200;
    J=zeros(1,Nu);               % array para guardar
       distancias
    h= waitbar(0,'Calculating');
    for n= 1:Nu
        waitbar(n/Nu,h);
        if user ~= n
            J(1, n) = sum(usersByMovie(user,:)~=
               usersByMovie(n,:))/K;
        end
    end
    delete(h)
    %ordena
    [~, sortedJ] = sort(J);
    %o primeiro elemento e o propeio logo da zero e n
       conta
    %escolhe os 3 mais pequenos e mais similares

    SimilarUsers = [sortedJ(2) sortedJ(3) sortedJ(4)];

    moviesToPrint = [];
    for i = 1: length(SimilarUsers)
        for n = 1:length(Set{SimilarUsers(i)})
            if (~ismember(Set{SimilarUsers(i)}(n), Set
               {user})) && ~ismember(Set{SimilarUsers(
               i)}(n), moviesToPrint)
                moviesToPrint = [moviesToPrint Set{
                   SimilarUsers(i)}(n)];
            end
        end
    end

    if ~isempty(moviesToPrint)
```

```
        for i = 1:length(moviesToPrint)  % Display dos
            filmes sugeridos
           fprintf('%s , ID:%d\n', listMovies{
              moviesToPrint(i)}, moviesToPrint(i));
       end
    else
        fprintf('\nNao Existe Filmes\n');
    end
    pause;
end
```

Explicação

Numa primeira fase será calculado a distância de jaccard para todos os users em relação ao user atual, de seguida as distâncias serão ordenadas e os users com as distâncias mais pequenas serão selecionados e passaram por um loop para que os seus filmes possam ser disponibilizados ao user.

Capítulo 5

Opção 3

A 3 opção ira listar os 2 filmes com nota superior ou igual a 3 e que são similares ao filmes do user, esta comparação e feita em relação os géneros de cada filme.

5.1 dados.m

```
%% moviesByGenero

K = 200;  % Numero de funcoes de dispersao
moviesByGenero = inf(length(listMovies),K);
for j = 1:length(listMovies)
    for l =2 : length(listMovies(j,:))
        if ~isa(listMovies{j,l}, "double")
            chave = char(listMovies{j,l});
        end
        hash = zeros(1,K);
        for kk = 1:K
            chave = [chave num2str(kk)];
            hash(kk) = DJB31MA(chave,127);
        end
        moviesByGenero(j,:) = min([moviesByGenero(j,:)
           ; hash]);  % Valor minimo da hash para este
            genero
    end
end

fprintf("Genero Done!\n");
```

Explicação

Aqui será feito o mesmo calculo que em 4.1, no entanto neste caso foram calculadas as *Hash* dos filmes em relação ao seu género.

5.2 app.m

```
%% movieGenreSugestions
function movieGenreSugestions (Nm, moviesByGenero,
    user, Set, listMovies)
     K = 200;
     count = 0;
     h= waitbar(0,'Calculating');
     for n1=1: length(Set{user})
         waitbar(n1/length(Set{user}),h);
         if(Set{user}(n1,2) >= 3)
             count = count + 1;
             temp = [];
             n1 = Set{user}(n1,1);
             J{count, 1} = n1;
             for n2= 1:Nm
                 if(n2 ~= n1 && ~ismember(n2,Set{user
                     }(:,1)))
                      jaccard = sum(moviesByGenero(n1,:)
                         ~=moviesByGenero(n2,:))/K;
                      if(jaccard <= 0.9)
                          temp = [temp n2];
                      end
                 end
             end
             J{count, 2} = temp;
         end
     end

     delete(h)


     counter = zeros(1,Nm);
     for h = 1: Nm
         for j=1:length(J)
             if(ismember(h, J{j, 2}))
                 counter(:,h) = counter(:,h) + 1;
             end
         end
     end
```

```
    %ordena
    [~, sortedJ] = sort(counter);
    %o primeiro elemento e o propeio logo da zero e n
       conta
    %escolhe os 2 mais pequenos e mais similares
    nMovies = 2;
    for i = 0: nMovies-1
        fprintf('%s , ID:%d\n', listMovies{sortedJ(end
            - i)}, sortedJ(end - i));
    end
    pause;
end
```

Explicação

Aqui será calcula a distância de jaccard para todos os filmes do user em questão, em relação aos filmes todos. Filtrando os filmes com distância maior que 0.9 e com avaliação inferior a 3,por esta razão foi necessário criar uma variável extra chamada de *count* pois com a eliminação de filmes indesejados *n1* não será continuo. De seguida os filmes que não são descartados serão inseridos num conjunto por cada filme do user e depois e contado o número de vezes que cada um aparece em todos os conjuntos. Os 2 filmes que aparecerem em mais conjuntos serão apresentados ao utilizador.

Capítulo 6

Opção 4

Nesta opção o user terá de fornecer um string com um nome de um filme ou similar, a aplicação ira retornar os filmes com os títulos mais similares ao string fornecido.

6.1 dados.m

```
%% usersByMovie Shingles

shingleSize = 3;
K = 200;  % Numero de funcoes de dispersao
textShingles = inf(length(listMovies),K);
for j = 1:length(listMovies)
    word = lower(listMovies{j,1});
    shingles = {};
    for i = 1: length(word) - shingleSize +1
        shingles{i} = word(i:i+shingleSize-1);
    end

    for i =1 :length(shingles)
        chave = char(shingles{i});
        hash = zeros(1,K);
        for kk = 1:K
            chave = [chave num2str(kk)];
            hash(kk) = DJB31MA(chave,127);
        end
        textShingles(j,:) = min([textShingles(j,:);
           hash]);  % Valor minimo da hash para este
           shingle
    end
end
```

```
fprintf("Shingles Done!\n");
```

Explicação

Aqui estão a ser criados shingles para todos os títulos dos filmes para mais tarde poderem ser comparados com a pesquisa do utilizador. Optamos por um Size de 3 para os shingles pois os títulos são pequenos e porque assim serão criados mais "padrões"o que fará com que a pesquisa seja mais eficiente. O ultimo loop irá percorrer todos os filmes e calcular um *Hash* para todos os filmes com base nos seus shingles.

6.2 app.m

```
%% titleSearch
function titleSearch (Mn, textShingles, listMovies)

    title = lower(input("Type a Movie: ", "s"));
    shingleSize = 3;
    K = 200;
    shingles = {};
    for i = 1: length(title) - shingleSize +1
        shingles{i} = title(i:i+shingleSize-1);

    end


    usersByMovieFind = inf(1,K);
    for i =1 :length(shingles)
        chave = char(shingles(i));
        hash = zeros(1,K);
        for kk = 1:K
            chave = [chave num2str(kk)];
            hash(kk) = DJB31MA(chave,127);
        end
        usersByMovieFind(1,:) = min([usersByMovieFind
           (1,:); hash]);  % Valor minimo da hash para
            este genero
    end

    flag = false;
    J=zeros(1,Mn);
    for n= 1:Mn
        J(1, n) = sum(textShingles(n,:)~=
           usersByMovieFind(1,:))/K;
```

```
    end

    [~, sortedJ] = sort(J);
    for j= 1: 5
        if~(J(1,sortedJ(j)) == 1)
            flag = true;
            fprintf("%s, ID:%d\n", listMovies{sortedJ(
               j)}, sortedJ(j));
        end
    end

    if(~flag)
        fprintf("Pesquisa Impossivel\n");
    end
    pause;
end
```

Explicação

Aqui foi pedido ao user um input depois e calculado os shingles do mesmo e o seu *Hash*, para de seguida ser calculado a distância de jaccard entre o *input* e todos os filmes, os 5 filmes com menor distância de jaccard são de seguida apresentados ao user por ordem crescente. Para que isto seja possível, este percorreu se as lista J e foi avaliado se o filme tiver distância de 1 este não e similar ao text logo e descartado, caso não exista nenhum filme similar o user recebe a mensagem não há filmes similares.

Capítulo 7

App

```
%% APP
function appFuntion(user, users, Set, listMovies,
   usersByMovie, moviesByGenero, textShingles, BF)
    while(1)
        clc;
        if (user == 0)
            tmp = input('Insert User ID (1 to 943): ')
               ;
            if (tmp< 1 || tmp>943)
                fprintf('User ID must be between 1 and
                    943!');
                pause(2);
            else
                user = tmp;
            end
        else
            option = input(['1 - Your Movies' ...
                            '\n2 - Suggestion of
                               movies based on other
                               users' ...
                            '\n3 - Suggestion of
                               movies based on already
                                evaluated movies' ...
                            '\n4 - Search Title' ...
                            '\n5 - Exit' ...
                            '\nSelect choice: ']);

            switch option
                case 1
```

```
                    fprintf("Option 1 was selected!\n
                        ");
                    pause(1);
                    yourMovies(user, Set, listMovies,
                        BF);
                case 2
                    fprintf("Option 2 was selected!\n
                        ");
                    pause(1);
                    otherUserSugestions (length(users
                        ), usersByMovie, user, Set,
                        listMovies);
                case 3
                    fprintf("Option 3 was selected!\n
                        ");
                    pause(1);
                    movieGenreSugestions (length(
                        listMovies), moviesByGenero,
                        user, Set, listMovies)
                case 4
                     fprintf("Option 4 was selected!\
                         n");
                     pause(1);
                     clc;
                     titleSearch(length(listMovies),
                         textShingles, listMovies)
                case 5
                    return
            end
            if(option<=5 && option >=1)
                user = 0;
            end
        end
    end
end
```

Explicação

Aqui será onde a app ira decidir o que fazer com base nos *inputs* do user, esta começa por pedir o ID do user, validando se este se encontra entre 1 e 943. De seguida e pedido ao user que escolha uma opção, e Este loop e repetido até ser presionado a opção 5.

Capítulo 8

Conclusão

Para o Desenvolvimento da app tivemos que usar, múltiplos conceitos explorados nas aulas, também foi necessário desenvolver novas funções com base no que foi criado nas aulas como o exemplo do Counting Bloom Filter que teve a base do Bloom Filter desenvolvido na aula, outros exemplos são as MinHash que tiveram de ser continuamente adaptadas para obter o objetivo pretendido. E necessário destacar que usamos apenas a função "DJB31MA.m", e que não necessitamos de funções de dispersão, como *hashstring* ou string2hash. Optamos pelos valores de K = 200 pois este valor irá obter valores de similaridade bastante próximo do real, apesar de este valor ser elevado e demorar a ser calculado sentimos que não afetaria bastante a *app* , uma vez que o Script "dados.m"só será executado periodicamente. A app irá fornecer dados com mais precisão, para alem disso como os dados são mais precisos a app irá demorar menos tempo a fornecer valores uma vez que esta tem menos valores para percorrer.