Relatório MPEI

Guião PL4 - Algoritmos Probabilísticos

Universidade de Aveiro

Métodos Probabilísticos para Engenharia Informática
Departamento de Eletrónica, Telecomunicações e Informática

Ano letivo 2022/2023

Turma P3

Marco Almeida - 103440

Rui Machado - 65081

Introdução

Este trabalho consistiu no desenvolvimento duma aplicação em MATLAB para simular um sistema de disponibilização de filmes. Foram-nos fornecidos 2 ficheiros de dados, o primeiro "films.txt" contém uma lista de títulos de filmes, assim como um conjunto de géneros que o identificam. O segundo é uma base de dados com todos os filmes que os utilizadores avaliaram.

O enunciado exigia que fossem desenvolvidos 2 scripts, um deles "Suporte.m" para ler os ficheiros fornecidos e criar todas as estruturas necessárias para a execução da aplicação e o outro "app.m" para correr a aplicação e chamar todas as funções necessárias.

Ao longo deste relatório definimos como *user* o utilizador que está a usar a aplicação para distinguir dos restantes utilizadores, com os quais vão ser feitas comparações de similaridades.

Suporte

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
udatax = load('u.data');                                 % Dados de users
films = readcell('films.txt', 'Delimiter', '\t');    % Lista de Filmes

% Udata fica apenas com as duas primeiras colunas
udata = udatax(:,1:2);

% Lista de utilizadores
users = unique(udata(:,1));    % Extrai os IDs dos utilizadores
numberOfUsers = length(users); % Numero de utilizadores

% Constroi estrutura para armazenar a lista de filmes por utilizador
moviesByUser = cell(numberOfUsers,1);
% Para cada utilizador Obtem os filmes de cada um
% e insere-os na estrutura
for i = 1:numberOfUsers
    lines = find(udata(:,1) == users(i));
    moviesByUser{i} = [moviesByUser{i} udata(lines,2)];
end

K = 100; % Numero de funçoes de dispersao
MinHashOP2 = inf(numberOfUsers,K);

% Para cada utilizador, faz-se MinHash ao conjunto de filmes vistos por ele
h= waitbar(0,'Calculating MinHash Op2');
for i = 1:numberOfUsers
    waitbar(i/numberOfUsers,h);
    setOfMovies = moviesByUser{i};

    for j = 1:length(setOfMovies)
        chave = char(setOfMovies(j));
        hash = zeros(1,K);
        for z = 1:K
            chave = [chave num2str(z)];
            hash(z) = DJB31MA(chave,127);
        end
        % Valor minimo da hash para este filme%
        MinHashOP2(i,:) = min([MinHashOP2(i,:); hash]);
    end
end
delete (h)

% Constroi estrutura para armazenar a lista de géneros por filme
genresByFilm = cell(length(films),1);
% Para cada filme Obtem os géneros de cada um
% e insere-os na estrutura
for i = 1:length(films)
    genresByFilm{i} = films(i,2:7);
end

K = 100; % Numero de funçoes de dispersao
MinHashOP3 = inf(length(films),K);
```

```
h= waitbar(0,'Calculating MinHash Op3');
for i = 1:length(films)
    waitbar(i/length(films),h);
    setOfGenres = genresByFilm{i};

    for j = 1:length(setOfGenres)
        % Cell pode estar vazia devido ao numero de géneros não ser
        % constante por filme
        % Se cell for vazia não é considerada para o MinHash
        if ismissing(setOfGenres{j}) == 0
            chave = char(setOfGenres{j});
            hash = zeros(1,K);
            for z = 1:K
                chave = [chave num2str(z)];
                hash(z) = DJB31MA(chave,127);
            end
            % Valor minimo da hash para este filme
            MinHashOP3(i,:) = min([MinHashOP3(i,:);hash]);
        end
    end
end
delete (h)

shingle_size = 3; % Tamanho do shingle
K = 100;          % Numero de funcoes de dispersao
MinHashOP4 = inf(length(films),K);

h = waitbar(0,'Calculating MinHash Op4');
for i = 1:length(films)
    waitbar(i/length(films),h);

    titulo = lower(films{i,1});
    shingles = {};
    % Criacao de shingles para cada filme
    for j = 1:length(titulo) - shingle_size + 1
        shingle = titulo(j:j+shingle_size-1);
        shingles{j} = shingle;
    end

    for j = 1:length(shingles)
        chave = char(shingles(j));
        hash = zeros(1,K);
        for z = 1:K
            chave = [chave num2str(z)];
            hash(z) = DJB31MA(chave,127);
        end
        % Valor minimo da hash para este shingle
        MinHashOP4(i,:) = min([MinHashOP4(i,:);hash]);
    end
end
delete(h);

m = length(films);
n = 8 * length(films);
k = 3;

BloomFilter = zeros(1,n);
h= waitbar(0,'Calculating Bloom Filter');
for i = 1:length(udatax)
    waitbar(i/length(udatax),h);
    movieElem = udatax(i,2);

    if udatax(i,3) >= 3
        for j = 1:k
            movieElem = [movieElem num2str(j)];
            hash = DJB31MA(movieElem, 127);
            hash = mod(hash,n) + 1;
            BloomFilter(hash) = BloomFilter(hash) + 1;
        end
    end

end
delete(h);
```

O ficheiro de Suporte serve para ler os ficheiros de entrada e criar as estruturas de dados necessárias à aplicação. Neste ficheiro criamos um vetor MinHash2, usado na opção 2 do menu, que contém a assinatura dos conjuntos de filmes de cada utilizador. Seguidamente criamos o MinHash3, que contém a assinatura dos conjuntos de géneros para cada filme.

Decidimos usar K = 100 nas funções de dispersão usadas nos vetores MinHash pelos resultados obtidos na execução do guião prático nas aulas anteriores que nos levaram à conclusão que este número de funções de dispersão era um bom compromisso entre tempo de execução e proximidade do valor real de similaridade.

Para a opção 4 desenvolvemos o MinHash4 que contém a assinatura dos conjuntos de shingles criados para cada filme. Os shingles são criados a partir dos títulos de filmes. Mais uma vez optámos por usar shigles_size = 3 porque achámos que seria um bom compromisso entre tempo de execução e performance.

Ainda para dar suporte à opção 4, criamos um vetor BloomFilter para fazer a contagem do número de vezes que um filme é avaliado com nota igual ou superior a 3.

Após testarmos o programa considerámos que os valores escolhidos produziam bons resultados.

O Ficheiro suporte guarda também outras variáveis úteis às funções como o número de utilizadores, um array com os conjuntos de filmes por utilizador e outros.

Main

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
films = readcell('films.txt', 'Delimiter', '\t');
data = load('u.data');

userId = getID();
op = getChoice();
while op ~= 5

    switch op
        case 1
            watchedMovies = getMoviesAndID(userId);
        case 2
            recByUser = getRecByUser(userId);
        case 3
            recByGenre = getRecByGenre(userId);
        case 4
            movieTitles = searchTitle();
        case 5
            return
        otherwise
            disp('Invalid option')
    end

    op = getChoice();

end
```

O main é o script que corre o programa, apenas chama as várias funções necessárias à execução. A função getID() pede um ID ao *user* e a função getChoice() apresenta um menu de escolha ao *user* para este escolher uma funcionalidade do programa. Estas funcionalidades são implementadas nas restantes funções que serão mencionadas nos próximos capítulos.

Função GetMoviesAndID

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function movies = getMoviesAndID(ID)
    Suporte = load("Suporte.mat");
    udata = Suporte.udata;
    films = Suporte.films;

    movies = {};

    for i = 1:size(udata, 1)
        if udata(i, 1) == ID
            movies = [movies; [udata(i, 2), films(udata(i, 2), 1)]];
        end
    end

    % Display List of Movies
    fprintf("List of Movies You Watched:\n")
    for i = 1:length(movies)
        disp(movies(i,:))
    end

end
```

Esta função é chamada quando o *user* escolhe a opção 1 do Menu e tem como objetivo listar todos os filmes que o *user* já viu apresentando para cada filme o seu ID e título. Apenas tem como parâmetro de entrada o ID do *user*, visto que todos os outros dados necessários podem ser lidos do ficheiro Suporte criado anteriormente.

Função GetRecByUser

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function moviesFromBoth = getRecByUser(ID)
    Suporte = load("Suporte.mat");
    moviesByUser = Suporte.moviesByUser;
    numberOfUsers = Suporte.numberOfUsers;
    MinHashOP2 = Suporte.MinHashOP2;
    films = Suporte.films;

    distancesJ = ones(1, numberOfUsers);

    h = waitbar(0, 'Calculating Jaccard Distances');
    K = 100;
    for i = 1:numberOfUsers
        waitbar(i/numberOfUsers, h);
        if i ~= ID
            % Calculamos a distancia de Jaccard para todos os pares possiveis desse user
            distancesJ(i) = sum(MinHashOP2(i,:) ~= MinHashOP2(ID,:))/K;
        end
    end
    delete(h);

    % Procurar os 2 utilizadores com menor distancia de Jaccard
    [~, min1] = min(distancesJ);
    distancesJ(min1) = 1;
    [~, min2] = min(distancesJ);

    disp(min1)      % Imprime no Terminal
    disp(min2)      % o ID dos 2 utilizadores

    % Inserir num vetor os filmes dos 2 utilizadores encontrados

    A = cell2mat(moviesByUser(ID));         % A = Lista de filmes do utilizador
```

```
    B = cell2mat(moviesByUser(min1));       % B = Lista de filmes do minFirst
    C = intersect(A,B);                     % C = Filmes de Ambos
    moviesFromFirst = setdiff(B,C);         % moviesFromFirst = filmes de minFirst que
utilizador desconhece

    B = cell2mat(moviesByUser(min2));       % B = Lista de filmes do minFirst
    C = intersect(A,B);                     % C = Filmes de Ambos
    moviesFromSecond = setdiff(B,C);        % moviesFromSecond = filmes de minSecond que
utilizador desconhece

    moviesFromBoth = union(moviesFromFirst,moviesFromSecond);

    % Print List of Movies
    fprintf("List of Movies By Similar User:\n")
    for i = 1:length(moviesFromBoth)
        disp(films(moviesFromBoth(i)))
    end

end
```

Esta função é chamada quando o *user* escolhe a opção 2 do Menu. Tem como propósito encontrar os 2 utilizadores mais similares ao *user* e listar os filmes vistos por eles que o *user* ainda não tenha avaliado. Para isso, começamos por calcular as distancias de Jaccard entre o *user* e todos os outros utilizadores a partir das assinaturas criadas no MinHashOP2.

Depois disso procuramos os 2 utilizadores com as menores distâncias através da função min() do MATLAB.

Tendo os Ids dos utilizadores mais similares, obtemos os filmes vistos por eles através do moviesByUser (cell array com os conjuntos de filmes para cada utilizador) criado no ficheiro de suporte.

Para criar a lista final de filmes a mostrar, fazemos a interseção entre filmes vistos pelo *user* e os filmes vistos pelos utilizadores similares e usamos a função setdiff() do MATLAB entre a interseção gerada e os filmes dos utilizadores similares.

Função GetRecByGenre

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function recommendation = getRecByGenre(ID)
    Suporte = load("Suporte.mat");
    moviesByUser = Suporte.moviesByUser;
    films = Suporte.films;
    MinHashOP3 = Suporte.MinHashOP3;

    % Criar Lista dos Filmes que o Utilizador já avaliou
    moviesOfCurrentUser = moviesByUser{ID};
    numberOfMovies = length(moviesOfCurrentUser);

    SetOfSimilars = cell(numberOfMovies,1); % Estrutura para guardar conjuntos
    counter = zeros(1,length(MinHashOP3));  % contador dos conjuntos

    h = waitbar(0, 'Calculating Jaccard Distances');
    K = 150;
    for i = 1:numberOfMovies
        waitbar(i/numberOfMovies, h);
        movieID = moviesOfCurrentUser(i);

        for j = 1:length(MinHashOP3)
            distJ = sum(MinHashOP3(j,:) ~= MinHashOP3(movieID,:))/K;
            if distJ < 0.8
                if isempty(intersect(moviesOfCurrentUser,j)) % verificar que não pertence aos
                                                             % filmes vistos pelo utilizador
                    SetOfSimilars{i} = [SetOfSimilars{i} j];
                    counter(j) = counter(j) + 1;
                end
            end
        end
    end
    delete(h);

    % Procurar os 2 filmes com maior contagem
    [~, max1] = max(counter);
    counter(max1) = 0;
    [~, max2] = max(counter);

    fprintf("List of Films By Similar Genre:\n");
    disp(films(max1));
    disp(films(max2));

    recommendation = [max1 max2];

end
```

Esta função é chamada quando o *user* escolhe a opção 3 do menu. O intuito desta função é procurar, para cada filme visto pelo *user*, todos os filmes com uma distância de Jaccard (em termos de géneros cinematográficos) inferior a 0.8 e após esse processo perceber quais os 2 filmes que surgiram mais vezes.

Começamos por obter os filmes vistos pelo *user* a partir da variável moviesByUser do ficheiro Suporte.

Criamos um cell array SetOfSimilars para guardar os filmes que cumpram os requisitos mencionados anteriormente e um array counter para fazer a contagem dos filmes.

Por cada filme visto pelo utilizador, calculamos as distâncias de Jaccard com todos os outros através do MinHashOP3 criado no Suporte e sempre que essa distancia é inferior a 0.8 guardamos esse filme no SetOfSimilars (excluímos filmes que o *user* já tenha visto usando isempty() de uma interseção do filme com o conjunto de filmes do *user*). Sempre que um filme é guardado no SetOfSimilars incrementamos no counter a posição equivalente ao filme.

No fim deste processo usamos a função max() no counter para descobrir quais 2 filmes surgiram mais vezes. Os índices resultantes no counter são os equivalentes aos ids dos filmes.

Função SearchTitle

```
function movieTitles = searchTitle()
    Suporte = load("Suporte.mat");
    films = Suporte.films;
    MinHashOP4 = Suporte.MinHashOP4;
    BloomFilter = Suporte.BloomFilter;

    str = lower(input('\nWrite a String: ', 's'));
    shingle_size = 3;

    % Cell array com os shingles da string introduzida
    shinglesStr = {};
    for i = 1:length(str) - shingle_size + 1
        shingle = str(i:i+shingle_size-1);
        shinglesStr{i} = shingle;
    end

    K = 100;
    % Fazer a MinHash dos shingles da string introduzida
    MinHashString = inf(1,K);
    for j = 1:length(shinglesStr)
        chave = char(shinglesStr{j});
        hash = zeros(1,K);
        for z = 1:K
            chave = [chave num2str(z)];
            hash(z) = DJB31MA(chave, 127);
        end
        MinHashString(1,:) = min([MinHashString(1,:); hash]);
    end

    % Guardar Distancias de Jaccard entre a string e cada filme
    distancesJ = ones(length(films),1);
    h = waitbar(0,'Calculating Jaccard Distances');
    for i = 1:length(films)
        waitbar(i/length(films), h);
        distancesJ(i) = sum(MinHashOP4(i,:) ~= MinHashString)/K;
    end
    delete(h);

    % Procurar os 5 filmes com menor Distancia
    [~, min1] = min(distancesJ);
    distancesJ(min1) = 1;
    [~, min2] = min(distancesJ);
    distancesJ(min2) = 1;
    [~, min3] = min(distancesJ);
    distancesJ(min3) = 1;
    [~, min4] = min(distancesJ);
    distancesJ(min4) = 1;
    [~, min5] = min(distancesJ);
```

```
% Pesquisar o nr de vezes que cada filme teve avaliaçao >= 3
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
movieTitles = [min1 min2 min3 min4 min5];
countVector = zeros(1,5);

n = Suporte.n;
k = Suporte.k;

for i = 1:length(movieTitles)
    movieElem = movieTitles(i);
    minValue = inf;

    for j = 1:k
        movieElem = [movieElem num2str(j)];
        hash = DJB31MA(movieElem, 127);
        hash = mod(hash,n) + 1;
        count = BloomFilter(hash);
        if count < minValue
            minValue = count;
        end
    end
    countVector(i) = minValue;
end
```

A função SearchTitle é chamada quando o utilizador escolhe a opção 4 do Menu. Esta função não requer nenhum argumento de entrada, pois o seu propósito é encontrar filmes os filmes com o título mais parecido à string introduzida pelo *user*.

Esta função começa por pedir ao *user* a string do título a pesquisar através de um input(). Após ter uma string, esta é convertida num conjunto de shingles de tamanho 3 e calculamos a assinatura MinHash para o conjunto produzido. O tamanho dos shingles e o processo de obtenção da MinHash têm de ser iguais aos usados para o MinHashOP4 no ficheiro de Suporte

Depois disso calculamos as distancias de Jaccard entre a assinatura criada e todas as outras assinaturas correspondentes aos títulos de filmes da aplicação através do MinHashOP4 e obtemos os 5 filmes com as menores distancias através da função min().

Para cada um dos 5 filmes encontrados repetimos o processo de hash usado na criação do BloomFilter e guardamos o menor valor encontrado, esse valor deve corresponder ao número de vezes que o filme teve avaliação maior ou igual a 3.