Relatório - MPEI

Universidade de Aveiro(*UA*)

Eduardo Lopes Ferreira,
João Afonso Pereira Ferreira

VERSAO 1

Relatório - Avaliação

Métodos Probabilísticos para Engenharia Informática (MPEI)

Carlos Bastos

DETI - Departamento de Electrónica, Telecomunicações e Informática

Engenharia de Computadores e Telemática (ECT)

Eduardo Lopes Ferreira,
João Afonso Pereira Ferreira

(102648) edu.fernandes@ua.pt,
(103037) ferreiraafonsojoao@ua.pt

03/01/2023

Índice

Lista de Figuras

Capítulo 1

Introdução

Começou-se por criar 2 ficheiros, um para base e carregamento de dados e um para o desenvolvimento da aplicação, denominados *'data.m'* e *'application.m'*, respetivamente. Desenvolveu-se o código no ficheiro data remetente ao carregamento dos filmes, categoria, ano e avaliação, armazena ainda os filmes vistos por cada utilizador pelo seu id e ainda se implementou uma função *minHash* para cada título de filme e para cada shingle. O outro *script* é, de facto, a aplicação em si e o menu de opções carregadas com as variáveis criadas previamente no ficheiro data.

No diretório onde se encontra o código relativo ao teste, também se encontram os ficheiros:

- *"u.data"*, *"films.txt"*, *"DJB31MA.m"* e uns outros ficheiros com terminação em '.mat', onde são utilizados na aplicação *"application.m"*
- Deve-se ter com conta ainda que algum código é reutilizado dos guiões práticos realizados nas aulas práticas da Unidade Curricular (UC)

Capítulo 2

Desenvolvimento

2.1 Análise do ficheiro data.m

De seguida, nesta secção, será apresentado o código correspondente ao ficheiro *"data.m"* (2.1).

Como é possível observar, começou-se por criar um 'dic' de forma a percorrer as linhas do ficheiro *"films.txt"*, fazer *'load'* do ficheiro *"u.data"*.

De seguida, teve que se definir um valor k para a *minHash* dos **filmes**, pelo que, se decidiu atribuir o valor de 100, uma vez que desta forma é possível obter um cálculo de distâncias mais rápido na aplicação e ainda obter similaridades mais próximas do valor atual.

Para o valor de k para a *minHash* dos ***shingles***, decidiu-se que o valor de k teria que assumir o valor de 150, por ser mais eficaz na obtenção de resultados nas funções usadas na aplicação.

Utilizou-se ainda dois métodos para **guardar** variáveis definidas em *'data.m'*, por exemplo "save 'users.mat' Set; save 'movies.mat' dic "ou então "save 'data' 'MinHashValue' 'MinHashSig' "

```
dic = readcell('films.txt', 'Delimiter','\t');

u_data = load('u.data'); % loads file
u = u_data(1:end,1:2);
clear u_data;


users = unique(u(:,1)); % get users id
No = length(users); % numb of users
save 'noUsers.mat' No

Set = cell(No,1); % use cells
for n = 1:No
    ind = find(u(:,1) == users(n));
    Set{n} = [Set{n} u(ind,2)];
end

% saves set and dic
save 'users.mat' Set
save 'movies.mat' dic


K = 100;  % number of 'funções de dispersão'
MinHashValue = inf(No,K);
for i = 1: No
    group = Set{i};
    for j = 1:length(group)
        chave = char(group(j));
        hash = zeros(1,K);
    for kk = 1:K
        chave = [chave num2str(kk)];
        hash(kk) = DJB31MA(chave,127);
    end
    MinHashValue(i,:) = min([MinHashValue(i,:); hash]);  % min value of the hash
for a title
    end
end

shingle_size=3;
K = 150; % number of ;funções de dispersão'

MinHashSig = inf(length(dic),K);
for i = 1:length(dic)
    group = lower(dic{i,1});
    shingles = {};
    for j = 1 : length(group) - shingle_size+1  % shingles
        shingle = group(j:j+shingle_size-1);
        shingles{j} = shingle;
end

for j = 1:length(shingles)
    chave = char(shingles(j));
    hash = zeros(1,K);
    for kk = 1:K
        chave = [chave num2str(kk)];
        hash(kk) = DJB31MA(chave,127);
    end
    MinHashSig(i,:) = min([MinHashSig(i,:);hash]);  % min hash Value min for this
shingle
    end
end

save 'data' 'MinHashValue' 'MinHashSig'
```

Figura 2.1: data *"data.m"*, ficheiro que carrega os ficheiros fornecidos

2.2 Análise do ficheiro application.m

Nesta secção, será apresentado o código correspondente ao ficheiro *"application.m"* (2.2). Na figura 2.2, é possível constatar a forma como foram carregadas as variáveis e os dados do ficheiro data, bem como o pedido do ID ao utilizador variando entre 1 e 943.

```
clc;
clear all;

load users.mat;
load movies.mat;
load noUsers.mat;
load eval.mat;
load data;

% ask users id and validate it
id = 0;
fprintf(2,"Message: User's ID must be an integer!\n\n")
while id < 1 || id > 943
    id = input('Insert User ID (1 to 943): ');
    if id < 1 || id > 943
        fprintf("\nUser ID must be between 1 and 943!\n");
    end
end
```

Figura 2.2: Implementação do *load* dos dados e ID dos utilizadores

De seguida, optou-se pela construção do menu de opções, aquilo que os utilizadores irão ver como entrada, assim como foi pedido para este trabalho. Observando a figura 2.3. Em primeiro lugar é se apresentado um menu e, de seguida, o sistema fica á espera de um *input* do utilizador entre as opções possíveis:

- Your Movies
- Suggestion of movies based on other users
- Suggestion of movies based on already evaluated movies
- Movies feedback based on popularity
- Exit

Tendo em conta a informação e enunciado fornecido, primeiramente, foi feita uma antevisão do possível output para cada opção e, a partir daí, foi-se implementando o código com recurso ao código desenvolvido nas aulas práticas.

```
% Menu
menu = '\n1 - Your Movies\n2 - Suggestions of movies based on other
users\n3 - Suggestion of movies based on already evaluated movies\n4 -
Movies based on popularity\n5 - Exit\nChoice: ';
choice = input(menu);


% while choice is different from 5
while choice ~= 5
    if (choice < 1 || choice > 5)
           fprintf("\n%d is not a posible choice, try again:\n", choice)
           choice = input(menu);
    end
    switch choice
        case 1
            % ID's user movies
            fprintf("\nUser %d watched the following movies:\n\n",id);
            for i = 1:length(Set{id})
                fprintf('%d - %s\n', i, dic{Set{id}(i)});
            end
            fprintf(2, 'Press any key to continue.');
            pause; clc;
            choice = input(menu);% present menu
        case 2
            getUserSuggestions(No, MinHashValue, id, Set, dic);
            pause; clc;
            choice = input(menu);% present menu
        case 3
            getSuggestionEvMov(No, MinHashValue, id, Set, dic);
            pause; clc;
            choice = input(menu);% present menu
        case 4
            popularMovies(dic, MinHashSig, eval);
            pause; clc;
            choice = input(menu);% present menu
    end
end
```

Figura 2.3: Menu

```
Message: User's ID must be an integer!

Insert User ID (1 to 943): 123

1 - Your Movies
2 - Suggestions of movies based on other users
3 - Suggestion of movies based on already evaluated movies
4 - Movies based on popularity
5 - Exit
Choice:
```

Figura 2.4: Output

Após a execução do menu, passamos para as funções requisitadas, pelo que serão de seguida enumeradas.

2.2.1 Your Movies

```
Your Movies:

fprintf("\nUser %d watched the following movies:\n\n",id);
for i = 1:length(Set{id})
    fprintf('%d - %s\n', i, dic{Set{id}(i)});
end
fprintf(2, 'Press any key to continue.');
pause; clc;
choice = input(menu);% present menu
```

Figura 2.5: Your Movies

O código apresentado na figura 2.5 irá imprimir os filmes vistos por um determinado utilizador em forma de lista, sendo que o Set consta todos os ID dos filmes vistos e o dic contém todos os filmes disponíveis.

2.2.2 Suggestion of movies based on other user

A próxima função (getUserSuggestions) determina os 2 utilizadores mais similares ao utilizador ao utilizador atual e ainda apresenta as sugestões de títulos de filmes que foram avaliados por pelo menos um dos 2 utilizadores e que ainda não tenham sido avaliados pelo utilizador atual.

Como parâmetros escolhidos, tem-se o 'No' (número de utilizadores disponíveis), a minHashValue, esta contém os valores dispersão, o 'id' do utilizador, o Set consta todos os ID dos filmes vistos e o dic contém todos os filmes disponíveis.

Irá ser sugerido o filme utilizando o valor das minHash, calculou-se as distâncias de Jaccard usando k = 100. Para obter os ID dos utilizadores mais similares e para verificar se havia algum filme que o utilizador atual não tivesse visto e o similar sim, para isso utilizou-se a função proveniente do MATLAB 'ismember'.

Suggestion of movies based on other users:

```
            function getUserSuggestions(No, MinHashValue, id, Set, dic)
    fprintf('\nPosible genres of Movies:\n  - Adventure\n  - Action\n  -
Thriller\n  - Animation\n');
    fprintf('Calculating...\n')

    K = 100;  % mesmo número de funcoes de dispersão usados para a MinHash
na database
    J = ones(1, No);
    h = waitbar(0, 'Calculating...');
    for n = 1:No
        waitbar(n/No, h);
        if n ~= id
            J(n) = sum(MinHashValue(n,:) ~= MinHashValue(id,:))/K;  % dist
de Jaccard
        end
    end
    delete(h);

    [val1, SimilarUserId] = min(J);
    J(SimilarUserId) = inf;
    [val2, SimilarUserId2] = min(J);

    fprintf('\nMost similar user ID 1: %d\n', SimilarUserId);
    fprintf('\nMost similar user ID 2: %d\n', SimilarUserId2);

    sug = []; % suggestion
    for n = 1:length(Set{SimilarUserId})
        % if the similar user has a movie seen that current user hasn't
        if (~ismember(Set{SimilarUserId}(n), Set{id}))
            sug = [sug string(dic{Set{SimilarUserId}(n), 1})];
        end
    end

    for n = 1:length(Set{SimilarUserId2})
        % if the similar user has a movie seen that current user hasn't
        if (~ismember(Set{SimilarUserId2}(n), Set{id}))
            sug = [sug string(dic{Set{SimilarUserId2}(n), 1})];
        end
    end

    if isempty(sug)
        fprintf('\nNo movies that werent evaluated!\n');
    else
       fprintf('\nNot evaluated movie titles: \n');
       for i = 1:length(sug)  % print suggested movies
           fprintf(sug(i) + '\n');
       end

    end
    fprintf(2, 'Press any key to continue. ');
    pause;
    clc;
end
```

Figura 2.6: Suggestion of movies based on other users

2.2.3 Suggestion of movies based on already evaluated movies

Nesta sub secção, irá ser abordado, o procedimento na implementação da função getSuggestionEvMov, que, para cada filme já avaliado pelo utilizador ataul, a aplicação seleciona os filmes cuja distância estimada seja menor que 0.8 e que ainda não tenham sido avaliados pelo utilizador atual. Pelo que a função irá retornar os títulos dos filmes que aparecem no maior número de conjuntos.

Para isto, será passado como argumento o 'No' (número de utilizadores disponíveis), a minHashValue, esta contém os valores dispersão, o 'id' do utilizador, o Set consta todos os ID dos filmes vistos e o dic contém todos os filmes disponíveis.

No fim de utilizar a minHash e a distância de Jaccard, irá ordenar os filmes e depois escolhe os dois filmes mais similares ao utilizador atual.

```
%% GetSuggestionEvMovies
function getSuggestionEvMov(No, MinHashValue, id, Set, dic);
    K = 100;
    count = 0;

    for n1=1: length(Set{id})
        count = count + 1;
        aux = [];
        n1 = Set{id}(n1,1);
        J{count, 1} = n1;
        for n2= 1:No
            if(n2 ~= n1 && ~ismember(n2,Set{id}(:,1)))
                if n1 <= size(MinHashValue, 1)
                    jaccard =
sum(MinHashValue(n1,:)~=MinHashValue(n2,:))/K;
                    if(jaccard <= 0.8)
                        aux = [aux n2];
                    end
                end
            end
        end
        J{count, 2} = aux;
    end

    counter = zeros(1,No);
    for h = 1: No
        for j=1:length(J)
            if(ismember(h, J{j, 2}))
                counter(:,h) = counter(:,h) + 1;
            end
        end
    end

    [~, sortedJ] = sort(counter);
    % pick 2 more similar
    fprintf(dic{sortedJ(end)} + "\n");
    fprintf(dic{sortedJ(end-1)} + "\n");
    pause;
end
```

Figura 2.7: Suggestion of movies based on already evaluated movies

2.2.4 Movies feedback based on popularity

De seguida, foi pedido produzir uma função em que é esperado uma string introduzida pelo utilizador como forma de pesquisa de um determinado filme. A aplicação tem o dever de retornar os 5 nomes de filmes com títulos mais similares à string introduzida e, para cada nome, o número de vezes que esses filmes foram avaliados com notas superiores ou iguais a 3 (usando um *Counting Bloom Filter*)

Como argumentos, irão ser passados, o 'id' do utilizador, uma minHash para shingles, as 'grades' são carregadas no ficheiro 'films.txt' no ficheiro data: ´´´eval = u_data(1:end,2); clear u_data;´´´ para poder fazer a condição de apenas obter os filmes com avaliação superior ou igual a 3, o Set consta todos os ID dos filmes vistos e o BF (Couting Bloom Filter) é um tipo de filtro de *hash* que estima a contagem de elementos em um conjunto de dados e ainda é eficiente em termos de espaço de armazenamento e é rápido, mas tem uma precisão limitada.

```
function popularMovies(id, dic, MinHashSig, grades, Set, BF)
    str = lower(input('\nMovie: ', 's'));
    shingle_size = 3;  % Use the same number of shingles as in the database
    K = size(MinHashSig, 2);  % Use the same K as in the database for the movie title
shingles
    threshold = 3;


    % Cell array with shingles of the input string
    shinglesAns = {};
    for i = 1:length(str) - shingle_size+1
        shingle = str(i:i+shingle_size-1);
        shinglesAns{i} = shingle;
    end

    % Calculate MinHash signature for the input string
    MinHashString = inf(1,K);
    for j = 1:length(shinglesAns)
        chave = char(shinglesAns{j});
        hash = zeros(1,K);
        for kk = 1:K
            chave = [chave num2str(kk)];
            hash(kk) = DJB31MA(chave, 127);
        end
        MinHashString(1,:) = min([MinHashString(1,:); hash]);
    end

     % Distancia de Jaccard
    distJ = ones(1, size(dic,1));
    h = waitbar(0,'Calculating');
    for i=1:size(dic, 1)
        waitbar(i/K, h);
        distJ(i) = sum(MinHashSig(i,:) ~= MinHashString)/K;
    end
    delete(h);

    % Find the movie titles with the most similar MinHash signatures
    found = false;
    for i = 1:5
        [val, pos] = min(distJ);  % calculate min valor (+similar)
        if (val <= threshold)  %
            found = true;
            fprintf('%s   (%.2f) number of evaluations: %d\n', dic{pos, 1}, dic{Set{id}(i)},
insert(Set{id}(i), BF, threshold ) );
        end
        distJ(pos) = 3;  %  remove movie
    end
        % Find the MinHash signature with the smallest difference from the input string's
signature
        %diffs = sum(MinHashSig ~= MinHashString, 2);
        %[minDiff, minDiffPos] = min(diffs);
        % Calculate the number of times the movie was rated 3 or above
        %if minDiffPos > size(grades,2)
        %    return
        %end
       % numGrades3Plus = sum(grades(:,minDiffPos) >= 3);

    if (~found)
        fprintf('No movies found.\n');
    end
    %fprintf(2, 'Press any key to continue. ');
    pause;clc;

    % bloom filter function
    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; % 1-n
            BF(h) = BF(h) + 1;
        end
    end
end
```

Figura 2.8: Movies feedback based on popularity

Capítulo 3

Conclusão

Concluindo, foi necessário a reutilização de código previamente utilizado e adaptado ao nosso problema. Todas as opções do menu (Fig. 2.3) foram implementadas iterativamente, uma a uma, tentado ao máximo a otimização do código. Com este trabalho, foram desenvolvidos mais e melhor competências no contexto de *minHash, BF, distância de Jaccard*, e ainda uma melhor familiarização com a linguagem *MATLAB*. Conseguindo atingir os objetivos próximos do pretendido, damos por encerrado a realização do trabalho da unidade curricular (MPEI).

Acrónimos

ECT Engenharia de Computadores e Telemática

MPEI Métodos Probabilísticos para Engenharia Informática

UC Unidade Curricular