Ano Letivo 2020/2021

Métodos Probabilísticos para Engenharia Informática

Relatório da Secção para Avaliação 4

Turma P7

Realizado por:

Ricardo Antunes, 98275

Henrique Sousa, 98324

Conteúdo

Introdução

Este trabalho tem como objetivo pedir ao utilizador o seu id, aparecendo de seguida um menu, cujas opções são:

1. Listar os títulos dos filmes que o utilizador viu;
2. Pedir ao utilizador o género que lhe interessa e imprimir os filmes que o utilizador mais similar viu e que o utilizador atual ainda não. Caso não haja, dizer que não há nenhuma sugestão;
3. Pedir ao utilizador para inserir uma frase, apresentando os filmes com menor distância de Jaccard à frase introduzida;
4. Sair e terminar o programa;

```
Insert User ID (1 to 943): 24

1 - Your Movies
2 - Get Suggestions
3 - Search Title
4 - Exit
Select Choice:
```

Figura 1: Menu

Foram fornecidos os ficheiros u.data e u_item. O ficheiro u.data tem como objetivo fornecer os ids dos filmes que os utilizadores viram, estando estes valores nas duas primeiras colunas.

O ficheiro u_item dá-nos o nome do filme e os respetivos géneros. Por exemplo, se o utilizador nº 940 viu o filme cujo id é 4, então este estará na 4 linha do ficheiro u_item.

Estrutura do Código

Leitura dos ficheiros

Foi desenvolvido um Script ("Guiao4Leitura.m") para ler os ficheiros fornecidos. Neste script começámos por armazenar na variável "dic" o conteúdo do ficheiro 'u_item', onde ficarão armazenados todos os filmes existentes bem como os géneros em que os mesmos se enquadram e na variável 'udata' guardámos os dados do ficheiro 'u.data'. A variável "Set" irá armazenar os filmes vistos por cada um dos utilizadores. Para isto percorremos cada ID de utilizador e armazenámos os filmes visualizados da seguinte forma:

```
for n = 1:Nu  % Para cada utilizador
    % Obtém os filmes de cada um
    ind = find(u(:,1) == users(n));
    % E guarda num array. Usa células porque utilizador tem um número
    % diferente de filmes. Se fossem iguais podia ser um array
    Set{n} = [Set{n} u(ind,2)];
end
```

Figura 2: Criação dos Sets

Com os Sets formados para cada um dos 943 utilizadores já nos é possível criar a MinHash para os users com auxílio da função DJB31MA fornecida pelo docente com o seguinte código:

```
K = 50;
MinHashValue = inf(Nu,K);
for i = 1:Nu
    conjunto = Set{i};
    for j = 1:length(conjunto)
        chave = char(conjunto(j));
        hash = zeros(1,K);
        for l = 1:K
            chave = [chave num2str(l)];
            hash(l) = DJB31MA(chave, 127);

        end
        MinHashValue(i,:) = min([MinHashValue(i,:);hash]);
    end
end
```

Figura 3: Criação da MinHash para os Utilizadores

Falta-nos criar uma MinHash para os filmes. Para isso, recorremos a um raciocínio semelhante, mas desta vez com auxílio de shingles de 3 caracteres.

```
Nm = length(dic);
MinHashMovies=inf(Nm,K);
shinglesLength = 3;
for i = 1 : Nm
    filme = dic{i, 1};
    for j = 1 : length(filme)-shinglesLength+1
        shingle = lower(char(filme(j:j+shinglesLength-1)));
        h=zeros(1,K);
        for n = 1 : K
            shingle = [shingle num2str(n)];
            h(n)=DJB31MA(shingle,127);
        end
        MinHashMovies(i,:)=min([MinHashMovies(i,:);h]);
    end
end
```

Figura 4: Criação da MinHash para os filmes

Assim, guardámos todas as variáveis necessárias no ficheiro "ValoresGRD.mat" que será, posteriormente, lido pelo script principal.

```
save 'ValoresGRD.mat' dic users Nu Set MinHashValue MinHashMovies K
```

Figura 5: Armazenamento das variáveis no ficheiro 'ValoresGRD.mat'

Opção 1

A opção 1 lista os filmes visualizados pelo utilizador em questão. Para os listar desenvolvemos a função listMovies:

```
function listMovies(userID, Set, dic)
    userMovies = Set{userID}; %Dá o numero do filme visto pelo utilizador
    movies = strings(length(userMovies),1); %converter movies para um array de strings
    fprintf("\nFilmes Visualizados: \n");
    for i = 1 : length(userMovies) %Percorrer todos os numeros do filme ja visto pelo utilizador
       fprintf("- %s\n", dic{userMovies(i),1}); %transformar o numero associado ao filme para o respectivo nome
    end
end
```

Figura 6: Função 'listMovies'

Nesta função, começámos por guardar num cell array os ids dos filmes que o utilizador atual viu.

De seguida, fizemos um ciclo for com o objetivo de percorrer todos os filmes vistos, convertendo os ids dos filmes para os respetivos nomes, guardando-os num array de strings (movies). Para isso, fomos ao ficheiro 'u_item' e procurámos pela linha cujo número está associado ao id do filme. A primeira coluna desta linha fornece-nos o nome do filme em questão. Com isto é possível extrair e imprimir a lista de filmes visualizados pelo utilizador escolhido.

O output desta opção será algo deste género:

```
1 - Your Movies
2 - Get Suggestions
3 - Search Title
4 - Exit
Select Choice: 1

Filmes Visualizados:
- GoldenEye (1995)
- From Dusk Till Dawn (1996)
- Amityville: A New Generation (1993)
- 101 Dalmatians (1996)
- Operation Dumbo Drop (1995)
- Bastard Out of Carolina (1996)
- Children of the Corn: The Gathering (1996)
- Toy Story (1995)
- Sudden Death (1995)
- Silence of the Lambs, The (1991)
- Aristocats, The (1970)
```

Figura 7: Exemplo de output da opção 1

Opção 2

Na opção 2 é pedido para dar uma sugestão de filmes ao utilizador. Para isto, a aplicação deve primeiro pedir-lhe o género de filme pretendido.

```
1 - Your Movies
2 - Get Suggestions
3 - Search Title
4 - Exit
Select Choice: 2

1- Action, 2- Adventure, 3- Animation, 4- Children's5- Comedy, 6- Crime, 7- Documentary,
Select Choice: 3
```

Figura 8: Opção 2

De forma a determinar quais filmes sugerir ao utilizador, fomos encontrar o utilizador mais similar, através do minHash e da similaridade. De seguida, apresentámos os filmes com o género selecionado vistos pelo utilizador mais similar e não visualizados pelo utilizador atual.

```
1 - Your Movies
2 - Get Suggestions
3 - Search Title
4 - Exit
Select Choice: 2

1- Action, 2- Adventure, 3- Animation, 4- Children's5- Comedy, 6- Crime, 7- Documentary, 8- Drama
Select Choice: 2

Sugestões:
- Man Who Would Be King, The (1975)
- Jurassic Park (1993)
- Jumanji (1995)
- Ghost and the Darkness, The (1996)
- War, The (1994)
- Lost World: Jurassic Park, The (1997)
- Twister (1996)
- Abyss, The (1989)
- Wizard of Oz, The (1939)
- Stand by Me (1986)
- Escape from New York (1981)
- Great Escape, The (1963)
- Evil Dead II (1987)
- Dances with Wolves (1990)
```

Figura 9: Sugestões de filmes

Para solucionar este problema foi nos exigido que utilizássemos o método MinHash.

Começámos por criar a função similaridade, com o objetivo de criar um array, cujos valores irão ser a similaridade do utilizador em questão relativamente aos restantes.

```
function var = similaridade(userID, K, MinHashValue, Nu)
    var=zeros(Nu,Nu); % array para guardar a similaridade
    h= waitbar(0,'Calculating');
    for nl= 1:Nu
        waitbar(nl/Nu,h);
        if(nl ~= userID)
            var(nl) = (sum(MinHashValue(nl,:) == MinHashValue(userID,:))/K);
        else
            var(nl) = 0;
        end
    end
    delete (h)
end
```

Figura 10: *Função similaridade*

Assim, o valor máximo encontrado neste array corresponde ao utilizador mais similar.

Definimos o utilizador mais similar como user2.

```
[i,user2] = max(variavel);
```

Figura 11: Código para verificar o utilizador mais similar

De seguida, fizemos uma função para verificar se há sugestões de filmes com o género que o utilizador escolheu.

```
x = Set{user2};  %Filmes do utilizador mais similar
y = Set{userID}; %Filmes do utilizador atual
Printsugestoes(x, userGenre, dic,y) %funcao para pintar sugestoes
```

Figura 12: Criação de variáveis

Nesta função, percorremos todos os filmes do user2. Em primeiro lugar, vimos se cada filme do user2 pertence aos filmes vistos do utilizador atual. Caso não pertença, significa que ainda não foi visto pelo utilizador, e caso o seu género for o escolhido pelo utilizador imprimimos o nome do filme.

No caso dos filmes do user2 já terem sido todos vistos pelo utilizador ou não forem do género que o utilizador escolheu então não há sugestões a fazer, imprimindo uma frase a avisar do sucedido.

```
function Printsugestoes(x, userGenre, dic,y)
    tamanho = length(x);
    valor = 0;

    for i = 1: tamanho %percorrer todos os filmes do utilizador mais similar
        if x(i)~= y %Ver se os filmes do user + similar nao foram visto pelo user atual
            if dic{x(i), userGenre+2} == 1 % Ver se o filme é do género pretendido
                if valor == 0
                    fprintf("\nSugestões de filmes: \n");
                end
            fprintf("%s\n", dic{x(i), 1});
            valor = valor + 1;
            end
        end
    end
    %valor;
    if valor == 0 % %Se os filmes do user2 já foram vistos pelo user atual ou se
        fprintf("Não há sugestões"); % não são do género escolhido
    end
end
```

Figura 13: Função p*ara imprimir os filmes sugeridos*

Opção 3

Nesta opção, o utilizador pode pesquisar por um filme através do seu nome. A aplicação apresenta no máximo 5 títulos com a menor distância de Jaccard à string introduzida pelo utilizador. Esta opção não depende de qual utilizador a estiver a usar. Apenas são considerados filmes com distância de Jaccard menor ou igual a 0.99. Caso não seja encontrado nenhum título a aplicação deve informar o utilizador.

Para solucionar a opção 3 desenvolvemos uma função denominado de “searchMovie” com 6 argumentos: “userString” - String introduzida pelo utilizador; “shinglesLength” - Tamanho de cada shingle o qual definimos a 3; “dic” - cell array com informações dos filmes todos; “K” - número de hash functions a utilizar, tendo sido definido com valor 50; “MinHashMovies” - Matriz com os valores dos MinHashs dos filmes; “Nm” - Número total de filmes.

Para a variável K (número de hash functions) atribuímos o valor 50, uma vez que ao testarmos com outros valores, este foi o valor com os resultados mais realistas. Para os shingles, definimos a cada um deles três caracteres pela mesma razão.

```
function searchMovie (userString, shinglesLength, dic, K, MinHashMovies, Nm)
    limite=0.99;

    %Criar MinHash da String introduzida pelo utilizador
    MinHashString=inf(1,K);
    for j=1:length(userString)-shinglesLength+1
        shingle=lower(char(userString(j:j+shinglesLength-1)));
        h=zeros(1,K);
        for m=1:K
            shingle=[shingle num2str(m)];
            h(m)=DJB31MA(shingle,127);
        end
        MinHashString(1,:)=min([MinHashString(1,:);h]);
    end
    %

    %Armazenar os Titulos com distancia de Jaccard inferior ou igual a 0.99
    movies={};
    for n=1 : Nm
        distancia=1-sum(MinHashString(1,:) == MinHashMovies(n,:))/K;
        if distancia <= limite
            movies{end+1,1}= dic{n,1};
            movies{end,2}= distancia;
        end
    end
    %

    %Imprimir os resultados obtidos
    if length(movies) == 0
        fprintf("We couldn't find movies with that title\n");
    else
        movies = sortrows(movies, 2);
        for i = 1 : min(height(movies), 5)
            fprintf("- %s (Jaccard ~%.2f)\n", movies{i,1}, movies{i,2});
        end
    end
    %
end
```

Figura 14: Função searchMovie

Primeiramente, temos de criar uma Matriz com os valores do MinHash da String introduzida pelo utilizador. Para isso, usámos um raciocínio semelhante ao usado no script de leitura para a MinHashMovies, mas neste caso, apenas efetuámos o processo uma vez, visto que, apenas temos uma string.

```
MinHashString=inf(1,K);
for j=1:length(userString)-shinglesLength+1
    shingle=lower(char(userString(j:j+shinglesLength-1)));
    h=zeros(1,K);
    for m=1:K
        shingle=[shingle num2str(m)];
        h(m)=DJB31MA(shingle,127);
    end
    MinHashString(1,:)=min([MinHashString(1,:);h]);
end
```

Figura 15: Criação de MinHash para a string

De forma a extrair os filmes com distância de Jaccard menor ou igual a 0.99, criámos o cell array “movies”. Calculámos a distância entre a String e cada um dos filmes existentes através da MinHashString e da MinHashMovies. Se este valor for menor ou igual a 0.99 adicionamos o título à matriz e a sua respetiva distância de Jaccard.

```
movies={};
for n=1 : Nm
    distancia=1-sum(MinHashString(1,:) == MinHashMovies(n,:))/K;
    if distancia <= limite
        movies{end+1,1}= dic{n,1};
        movies{end,2}= distancia;
    end
end
```

Figura 16: Guardar os filmes com distância de Jaccard <= 0.99

Com isto feito, é possível agora tirar conclusões. Caso não haja títulos na matriz “movies” avisamos o utilizador que não foram encontrados filmes com a string introduzida. Caso contrário, imprimimos no máximo 5 filmes por ordem crescente de distância de Jaccard. Para isso, ordenámos a matriz por ordem crescente da segunda coluna e imprimimos os 5 primeiros títulos se o tamanho da matriz for igual ou superior a 5, senão imprimimos toda a matriz.

```
if length(movies) == 0
    fprintf("We couldn't find movies with that title\n");
else
    movies = sortrows(movies, 2);
    for i = 1 : min(height(movies), 5)
        fprintf("- %s (Jaccard ~%.2f)\n", movies{i,1}, movies{i,2});
    end
end
```

Figura 17: Imprimir os resultados

Foi necessário transformar os shingles para strings com apenas letras minúsculas, tanto para a criação da MinHashString como para a MinHashMovies com o objetivo de não haver distinção entre caracteres maiúsculos e minúsculos na pesquisa.

Todo este raciocínio resulta no seguinte output, introduzindo a string “die”:

```
Write a string: die
- Eddie (1996) (Jaccard ~0.92)
- Die Hard (1988) (Jaccard ~0.96)
- Die Hard 2 (1990) (Jaccard ~0.98)
- To Die For (1995) (Jaccard ~0.98)
- Marlene Dietrich: Shadow and Light (1996) (Jaccard ~0.98)
```

Figura 18: Exemplo de output para a opção 3

Opção 4

A opção 4 serve, simplesmente, para terminar o programa.

Conclusão

A realização deste trabalho permitiu-nos compreender temas como: MinHash, similaridade, distância de Jaccard e shingles. Consideramos mais adequado a utilização do valor 50 para a variável K (número de hash functions), uma vez que, os valores obtidos aproximaram-se mais à realidade.

Ao analisar os resultados obtidos deduzimos que o código foi bem concebido, podendo assim concluir que os objetivos deste trabalho foram realizados com sucesso.