2020/2021

MÉTODOS PROBABILÍSTICOS PARA ENGENHARIA INFORMÁTICA

Relatório PL04 – Turma P04

Prof. Amaro Sousa

Paulo Pereira (98430)

Luís Martins (98521

Índice

Introdução

Neste guião foi desenvolvida uma aplicação com algumas funcionalidades esperadas de um sistema online de disponibilização de filmes, essas funcionalidades são: a listagem de todos os filmes vistos por um utilizador, a listagem de sugestões de filmes de um género com base no utilizador mais similar ao atual e a opção de procurar por filmes similares a uma string introduzida.

Para atingir este objetivo foi criada uma interface de utilização da aplicação que de acordo com os inputs do utilizador realiza as funcionalidades propostas. Além disso foram tidos como base os ficheiros “u_item.txt” que contém a lista de todos os filmes e as suas classificações quanto aos géneros e o ficheiro “u.data.txt” do qual são usadas as primeiras duas colunas tendo assim associado a cada utilizador do sistema os filmes que este viu. Foi tido em conta também algum código e conclusões retiradas da parte sem avaliação deste guião.

Explicação das variáveis

K-número de funções de dispersão

Para o número de funções de dispersão, tendo em conta que vamos ter a criação de 3 métodos Min-hash sendo um deles durante a execução do programa decidimos usar um valor entre 100 e 200 como concluímos na parte sem avaliação, assim decidimos o valor 130 como o valor final pois achamos o tempo de execução e precisão no cálculo das distâncias de Jaccard aproximadas bastante satisfatório.

k-número de shingles

O número de shingles que pretendíamos seria entre 2-5 após a implementação da aplicação e várias pesquisas na opção 3 concluímos que 4 seria o valor desejado.

Estruturas

Para desenvolver a aplicação foram criados 2 scripts o primeiro dos quais "structmaker" cria as estruturas de dados associadas aos utilizadores e filmes, bem como a leitura dos ficheiros de entrada.

Leitura dos ficheiros de entrada

Código-

```
udata=load('u.data.txt');
movies= readcell("u_item.txt");
u= udata(1:end,1:2);
clear udata;
users = unique(u(:,1));
Nu= length(users);
m=length(movies);
for i=1:m
    movielist(i)=string(movies{i,1});
end
Set= cell(Nu,1);
for n = 1:Nu
    ind = find(u(:,1) == users(n));
    Set{n} = [Set{n} u(ind,2)];
end
```

Para inicializar o problema temos que ler os ficheiros "u_item.txt" e "u.data.txt" para isso usamos o código fornecido no guião, nesta parte inicial guardamos em dois arrays os títulos dos filmes e a lista de filmes vistos por cada utilizador.

Criação das Min-Hash

Para o desenvolvimento da aplicação foi necessário gerar duas estruturas pelo método Min-Hash a primeira das quais gera os vetores Min-Hash de cada utilizador e o segundo uma matriz com os vetores Min-Hash de cada filme. Na primeira é gerado para cada utilizador um vetor que para cada filme visto através da aplicação de 130 funções de dispersão, na segunda estrutura para cada filme são criados os shingles do seu título que depois são submetidos a 130 funções de dispersão.

Código-

```
K=130;
MinHashTable=inf(Nu,K);
for i=1:Nu %cria as Min-Hash
    conjunto=Set{i};
    for j=1:length(conjunto)
        chave=char(conjunto(j));
        h=zeros(1,K);
        for kk=1:K
            chave=[chave num2str(kk)];
            h(kk)=DJB31MA(chave ,127);
        end
        MinHashTable(i,:)=min([MinHashTable(i,:);h]);
    end
end

k=4; %Shingles
MinHashMovies=inf(m,K);
for i=1:m %cria as Min-Hash
    conj=char(lower(movies{i,1}));
    for j=1:length(conj)-k+1
        chave=conj(j:j+k-1);
        h=zeros(1,K);
        for kk=1:K
            chave=[chave num2str(kk)];
            h(kk)=DJB31MA(chave ,127);
        end
        MinHashMovies(i,:)=min([MinHashMovies(i,:);h]);
    end
end
```

Cálculo das distâncias aproximadas

Código-

```
for n1=1:Nu %Calcula as distâncias de Jaccard aproximadas
    for n2=n1+1:Nu
        J(n1,n2)= sum(MinHashTable(n1,:)~=MinHashTable(n2,:))/K;
    end
end
```

Após obtermos a estrutura com os vetores Min-Hash de cada utilizador foi necessário calcular as distâncias aproximadas de Jaccard para encontrar o utilizador mais similar a um outro utilizador.

Guardar o utilizador mais similar

Após o passo anterior precisamos de guardar os pares de utilizadores mais similar, feito da seguinte forma.

Código-

```
k=1;
SimilarUsers= zeros(1,3);
for n1= 1:Nu %Guarda numa matriz a distância entre 2 users de forma a que o par só apareça uma vez e não haja pares do tipo (1,1)
    for n2= n1+1:Nu
            SimilarUsers(k,:)= [users(n1) users(n2) J(n1,n2)];
            k= k+1;
    end
end
```

Aplicação

A aplicação inicia ao carregar as estruturas resultantes do script anterior e pede ao utilizador o id de o utilizador que este pretende ver. Para validar inputs do tipo inteiros criamos a função getValidInput.

Validação de inputs

Código-

```
function inp = getValidInput(prompt,error,min,max) %verifica os inputs
    while(1)
        inp = str2double(input(prompt, "s"));
        if (~isnan(inp) && (min <= inp && max >= inp))
            fprintf("\n");
            break;
        else
            fprintf(error);
        end
    end
end
```

Esta função aceita como argumentos o prompt, que indica ao utilizador o input necessário, o error, ou seja, a mensagem de erro que será mostrada quando o input for inválido, e dois valores o mínimo e máximo que definem entre que valores o input deve estar. A função em si verifica se é um número valido entre o máximo e o mínimo.

Menu

Após a validação é mostrado um menu com as diferentes opções, sendo a opção selecionada também validada com uso da função anterior.

Opção 1

Na primeira opção pretende se mostrar todos os filmes vistos por um utilizador foi desenvolvida uma pequena função que dá print a todos os títulos vistos pelo utilizador partindo das estruturas Set e movies.

Código-

```
function getUserMovies(user,Set,movies) %da print a todos os filmes do user selecionado
    size=length(Set{user});
    for id=1:size
       fprintf("%d - %s\n",id,movies{Set{user}(id),1});
    end
end
```

Opção 2

Na segunda opção pretende se que o programa sugira filmes de um determinado género, para isso é procurado o utilizador mais similar ao utilizador atual e são recomendados os filmes que este segundo utilizador viu que o primeiro não viu dentro de um determinado género, após verificação se o género selecionado é valido, foram criadas 3 funções a primeira getUserMovies tendo por base a função da primeira opção guarda os filmes de um género vistas por um user, a segunda getBestUser percorre a estrutura SimilarUsers e devolve o utilizador mais similar ao atual e por fim a terceira função getUnseenList compara os resultados da função getUserMovies para ambos utilizadores e dá print dos filmes não vistos pelo utilizador selecionado no início da aplicação.

Código-

```
function usermovies=getUserGenreMovies(user,Set,movies,genre)% devolve os filmes de um genero vistos por um user
    k=1;
    size=length(Set{user});
    for id=1:size
        if movies{Set{user}(id),genre}
           usermovies(k)=string(movies{Set{user}(id),1});
           k=k+1;
        end
    end
end
```

```
function BestUser=getBestUser(user,SimilarUsers) %devolve o utilizador mais similar
            x=1;
            for i=1:length(SimilarUsers)
                if SimilarUsers(i,1)==user
                    if SimilarUsers(i,3)<x
                        x=SimilarUsers(i,3);
                        BestUser=SimilarUsers(i,2);
                    end
                end
            end

end
```

```
function getUnseenList(UserMovies,BestUserMovies) %dá print da lista de filmes não vistos pelo user
            z=1;
            clear List;
            for i=1:length(BestUserMovies)
                control=0;
                for k=1:length(UserMovies) %para cada filme vai verificar se foi visto pelo user
                    if (BestUserMovies(i)==UserMovies(k))
                        control=1;
                    end
                end
                if control==0 % se control for 0 o User não viu o filme portanto guarda se o filme
                    List(z)=BestUserMovies(i);
                    z=z+1;
                end
            end
            for k=1:z-1 %print dos filmes
                fprintf("%d - %s\n",k,List(k));
            end
            if z==1 %caso não haja sugestões
                fprintf("No sugestions.\n")
            end
end
```

Opção 3

Na opção 3 pretende-se que após a introdução de um string por parte do utilizador sejam devolvidos títulos de filmes que sejam similares ao string introduzido. Para isso após a verificação se o tamanho do string tinha pelo menos o tamanho do shingle definido são chamadas duas funções, a primeira das quais gera as Min-Hash para o string introduzido e a segunda que com base nas duas estruturas Min-Hash(dos filmes e do string) dá print dos 5 filmes mais similares, caso não haja 5 são devolvidos todos os filmes similares.

Código-

```
function MinHashInput=makeMinHash(step,MinHashMovies,k)
            [ML,K]=size(MinHashMovies);
            MinHashInput=inf(1,K);
            for i=1:length(step)-k+1
                chave=step(i:i+k-1);
                h=zeros(1,K);
                for kk=1:K
                    chave=[chave num2str(kk)];
                    h(kk)= DJB31MA(chave,127);
                end
                MinHashInput(1,:)=min([MinHashInput(1,:);h]);
            end

end
function findMovies(MinHashInput,MinHashMovies,movielist)
            FindMovies=zeros(1,2);
            ct=1;
            [ML,K]=size(MinHashMovies);
            for n1= 1:ML
                 J= sum(MinHashInput(1,:) ~= MinHashMovies(n1,:))/K;
                 if J<=0.99
                        FindMovies(ct,:)= [J n1];
                        ct= ct+1;
                 end
             end
             FindMovies=sortrows(FindMovies,2);
             if sum(FindMovies)==0
                 fprintf("No titles found\n");
             else
                 for i=1:min([5 height(FindMovies)])
                     fprintf("%d - %s\n",i,movielist{FindMovies(i,2)});
                 end
             end
end
```

Opção 4

A opção 4 limita-se a quebrar o ciclo while dando return, terminando assim a aplicação.

Exemplos de outputs

Opção 1

```
1 - Mimic (1997)
2 - Ulee's Gold (1997)
3 - Incognito (1997)
4 - One Flew Over the Cuckoo's Nest (1975)
5 - Event Horizon (1997)
6 - Client, The (1994)
7 - Liar Liar (1997)
8 - Scream (1996)
9 - Star Wars (1977)
10 - Wedding Singer, The (1998)
11 - Starship Troopers (1997)
12 - Air Force One (1997)
13 - Conspiracy Theory (1997)
14 - Contact (1997)
15 - Indiana Jones and the Last Crusade (1989)
16 - Desperate Measures (1998)
17 - Seven (Se7en) (1995)
18 - Cop Land (1997)
19 - Lost Highway (1997)
20 - Assignment, The (1997)
21 - Blues Brothers 2000 (1998)
22 - Spawn (1997)
23 - Wonderland (1997)
24 - In & Out (1997)

Press any key to continue
```

Teste- 1 Filmes vistos pelo utilizador 4

Opção 2

```
No sugestions.

Press any key to continue
```

Teste- 2 Sugestões para a categoria 7 do utilizador 1

```
1 - Cape Fear (1962)
2 - Devil in a Blue Dress (1995)

Press any key to continue
```

Teste- 3 Sugestões para a categoria 10 do utilizador 1

Opção 3

```
1 - Usual Suspects, The (1995)
2 - White Balloon, The (1995)
3 - Rumble in the Bronx (1995)
4 - Birdcage, The (1996)
5 - Brothers McMullen, The (1995)

Press any key to continue
```

Teste- 4 Resultados da Opção 3 com introdução do string "the s"

```
1 - Scream (1996)
2 - Kicking and Screaming (1995)
3 - Screamers (1995)
4 - Scream 2 (1997)

Press any key to continue
```

Teste- 5 Resultados da Opção 3 com introdução do string "scre"