2022/2023 – Métodos Probabilísticos para Engenharia Informática

PL 4

Trabalho realizado por:

Bernardo Leitão da Costa Pinto (105926) – Turma P7

Maria João Machado Sardinha (108756) – Turma P7



# Índice

[Introdução 3](#_Toc124045162)

[Ferramentas 4](#_Toc124045163)

[- MinHash 4](#_Toc124045164)

[- Distância de Jaccard 4](#_Toc124045165)

[- Funções de Dispersão 4](#_Toc124045166)

[- Bloom Filter 5](#_Toc124045167)

[- Counting Bloom Filter 5](#_Toc124045168)

[Funções 6](#_Toc124045169)

[- main() 6](#_Toc124045170)

[- opcao1() 6](#_Toc124045171)

[- opcao2() 6](#_Toc124045172)

[- opcao3() 7](#_Toc124045173)

[- opcao4() 7](#_Toc124045174)

[Resultados 9](#_Toc124045175)

[Conclusão 12](#_Toc124045176)

[Fontes 13](#_Toc124045177)

# Introdução

Neste trabalho é nos pedido que tomemos em consideração um conjunto de dados sobre um conjunto de filmes e de utilizadores que avaliaram esses filmes e que, a partir destes, desenvolvamos uma aplicação.

Esta aplicação deverá começar por pedir o ID do filme ao utilizador e, após se verificar que este é válido, pede-se ao utilizador que escolha umas das opções do menu dado.

Para desenvolvermos a aplicação que nos era pedida usámos o MATLAB e aplicámos os conhecimentos aprendidos nesta cadeira (Métodos Probabilísticos para Engenharia Informática). Implementámos ainda algumas ferramentas importantes, tais como MinHash, Funções de Dispersão, Bloom Filter, …

Este relatório serve para explicar a utilização deste programa desenvolvido e para se entender o código usado na resolução de tal.

# Ferramentas

## **- MinHash**

MinHash é uma ferramenta muito importante na área da informatica usada para calcular a similaridade entre conjuntos.

A MinHash baseia-se em criar uma “assinatura” para cada conjunto e, posteriormente, calcula-se a similaridade entre essas duas assinaturas, calculado o número mínimo de elementos que um conjunto dever conter para que seja considerado um subconjunto do outro. Sendo que para a criação desta assinatura, tem de se pegar em cada “set” de informação e passar, através de uma função de dispersão, para cada um dos seus elementos. Depois, para cada “set” é verificado qual o mais pequeno dos valores obtidos (após o “hashing”) e será este valor que irá ser colocado na matriz das assinaturas.

## **- Distância de Jaccard**

A distância de Jaccard entre 2 conjuntos é calculada através da seguinte expressão: 1 – similaridade(set1, set2)

Sendo que a similaridade de Jaccard entre 2 conjuntos corresponde ao quociente entre o tamanho da interceção dos 2 conjuntos com o tamanho da união entre esses 2 conjuntos.

## **- Funções de Dispersão**

Uma função de dispersão é uma função cujo elemento de entrada é um conhecida como “key” (chave) (pode possuir qualquer tamanho) e o de saída corresponde a um “hash” (de tamanho). Este “hash” pode ser usado para aceder a uma estrutura de dados onde, na posição retornada, podem estar armazenadas diversas informações relativamente ao elemento de entrada.

Estas são usadas para facilitar o armazenamento e o acesso aos dados, ocupando muito menos espaço.

## **- Bloom Filter**

É uma ferramenta usada para testar se um dado elemento pertence a um conjunto.

Este filtro recorre a funções de dispersão, das quais resulta um número inteiro que, posteriormente, é usado para aceder a um vetor (o filtro). Todas as posições devolvidas pelas funções de dispersão são colocadas a 1 (sendo que este é inicializado com estas iguais a 0).

Após se verificar a presença de um dado elemento num dado conjunto, volta-se a passar pelas funções de dispersão e verifica-se se todos os valores das listas dos índices retornados são 1, para verificar se o resultado está correto.

## **- Counting Bloom Filter**

É uma variação do Bloom Filter tradicional, o qual permite a remoção de elementos do conjunto.

Nos Counting Bloom Filter cada valor no vetor é inicialmente definido como zeros e cada posição é associada a um contador. Quando ocorre a adição de um elemento ao conjunto, as funções hash são aplicadas ao elemento e os contadores correspondentes aumentam. Por outro lado, quando ocorre a remoção, os contadores diminuem e, caso este fique a corresponder a zero, indica que o elemento foi removido do conjunto.

# Funções

Aqui serão descritas de que se tratava cada função usada para que a aplicação funcionasse.

## **- main()**

Na main é onde o programa carrega os dados contidos no ficheiro “u.data” e “users.txt” e pede ao utilizador que este indique o ID do filme em questão e, caso este seja válido, aparece o menu. Caso não seja, o programa continua a pedir o ID do filme até que este seja válido.

No menu é pedido ao utilizador que escolha uma das opções dadas:

- caso escolha a “1” -> é reencaminhado para a função “opcao1”;

- caso escolha a “2” -> é reencaminhado para a função “opcao2”;

- caso escolha a “3” -> é reencaminhado para a função “opcao3”;

- caso escolha a “4” -> o programa termina

- caso escolha a outra -> é imprimida uma mensagem a indicar que é uma opção inválida e imprime, novamente, o menu e pede, novamente, a opção.

## **- opcao1()**

Nesta função são listados os nomes dos utilizadores que avaliaram o filme atual.

Para tal, são percorridas todas as linhas do ficheiro “u.data” e, caso o valor da segunda coluna (onde estão os ID’s dos filmes) dessa linha corresponda ao ID do filme dado pelo utilizador, o programa imprime o ID do utilizador dessa linha, assim como o seu nome e apelido. Sendo que para imprimir o nome e o apelido o programa vai buscar essa informação ao “dic” (que é onde está o conteúdo de “users.txt”), procurando na linha correspondente ao ID do utilizador a segunda e a terceira coluna (nome e apelido, respetivamente).

## **- opcao2()**

Na função opcao2() são determinados os 2 filmes mais similares ao filme atual (em termos de conjunto de utilizadores que avaliaram cada filme), apresentando os Ids e nomes dos utilizadores que nomearam pelo menos um dos filmes selecionados, mas que ainda não avaliaram o filme em questão.

Para que sejam determinados esses tais 2 filmes, começa-se por obter os conjuntos e Ids dos filmes, chamando a função createSet().

Depois é calculada a MinHash, inicializam-se as funções de dispersão, através da função inicFuncoesDispersao(), calcula-se as assinaturas usando a função calcularMatrizAssinaturas(), calcula-se as distâncias com base nessa matriz de assinaturas através da função calcularDistâncias() e determina-se os 2 pares similares com base na distância, através da função doissimilares().

De seguida percorrem-se as linhas do ficheiro “u.data” e vão se guardando os IDs dos utilizadores que já avaliaram o filme em questão numa variável e os IDs dos que já avaliaram os outros dois filmes encontrados noutras duas variáveis. Depois, percorrem-se esses dois vetores dos utilizadores que avaliaram esses tais dois filmes e verifica-se, para cada utilizador se esse é membro do vetor dos utilizadores que avalizaram o filme introduzido pelo utilizador e, caso não seja membro, adiciona-se esse utilizador a um vetor novo (que conterá os utilizadores todos pedidos nesta opção).

Por fim, imprimem-se os IDs dos utilizadores pedidos, assim como os seus nomes.

## **- opcao3()**

Nesta opção é nos pedido que, para cada utilizador que já avaliou o filme atual, a aplicação selecione os utilizadores cuja distância de Jaccard estimada, em termos de interesses, seja menor que 0.9 e que ainda não tenham avaliado o filme atual. Tal irá resultar num conjunto de potenciais utilizadores por cada avaliador do filme atual e no fim deverão ser impressos os IDs e nomes dos 2 utilizadores que aparecem no maior número de conjuntos.

Para tal, primeiro obtiveram-se, através da função createSet2, o conjunto de filmes por user e, por meio da função createSet, o conjunto de users por filme, uma vez que nesta opção, ambos os conjuntos serão relevantes.

Em seguida, é realizado, utilizando manipulação de matrizes e um ciclo for, o armazenamento dos interesses de cada user.

Depois, para calcular a distancia de jaccard através da MinHash, inicializam-se as funções de dispersão através da função inicFuncoesDispersao() e calcula-se as assinaturas, de acordo com os interesses, utilizando a função calcularMatrizAssinaturas().

Posteriormente, por meio da função calcularDistancias(), é realizado o calculo das distancias entre os users, e, além disso, como as assinaturas são passadas como parametros da função, as distancias estão diretamente relacionadas aos interesses.

Por fim, utilzando alguns ciclos for e estruturas condicionais, é realizada a filtragem dos users que, além de não terem avaliado o filme em questão, apresentaram uma distancia de jaccard menor do que 0.9 quando comparados aos users que avaliaram. Portanto, foi implementada uma tabela que, quando cada user cumpre as condições da estrutura condicional, o indice referente aquele user é incrementado em uma unidade, resultando, então, na quantidade de vezes que cada um cumpriu os requisitos da comparação. Finalmente, foi implementado o codigo responsável por extrair os dois users com mais ocorrencias e apresentar na tela.

## **- opcao4()**

Aqui, na opcao4() é pedido ao utilizador que insira, numa string, o nome de um filme (ou parte do nome) e o programa devolve 3 nomes de filmes com os títulos mais similares à string introduzida e, para cada nome, o número de vezes que este foi avaliado com uma nota superior ou igual a 3.

Para a realização desta opção começamos por pedir ao utilizador que inserir-se o nome do filme (ou parte dele) e carregámos os dados contidos no ficheiro “film\_info.txt”.

De seguida, percorremos linha por linha o ficheiro que continha as informações dos filmes e retirámos os nomes dos filmes, retirando também os espaços que este tinha a mais. Após termos os nomes dos filmes, com o auxílio da função getShingles(), calculamos os “shingles” para os nomes dos filmes. Depois, fomos verificar quais os “shingles” que ocorriam um maior número de vezes. Visto que o “shingle” que ocorria mais vezes seria um conjunto de espaços em branco, verificámos também qual o segundo, terceiro e quarto “shingle” que ocorriam mais vezes.

De seguida, percorremos novamente a lista de nomes dos filmes, para percebermos quais nomes de filmes é que continham esses tais “shingles” que ocorriam mais vezes e armazenámos esses nomes de filmes numa lista. Para fins informativos, imprimimos os “shingles” mais frequentes e fomos procurar os IDs dos filmes em questão.

Por fim, fomos verificar (através dos IDs dos filmes), quantas vezes cada um desses filmes foi avaliado com uma nota igual ou superior a 3 e imprimimos os resultados pedidos.

# Resultados



Figura 1 - Output opção 1 para o filme 57

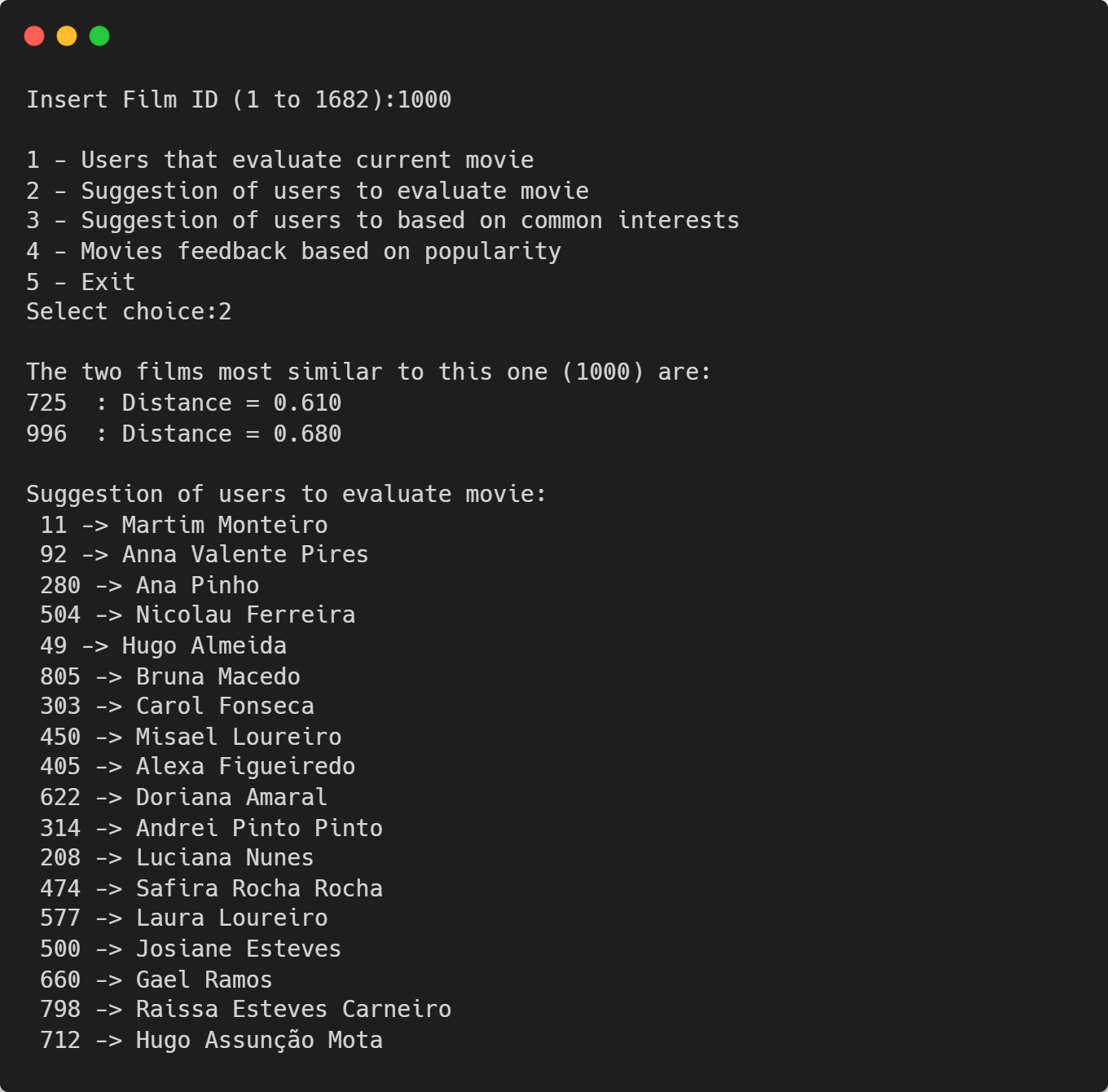


Figura 2 - Output opção 2 para o filme 1000

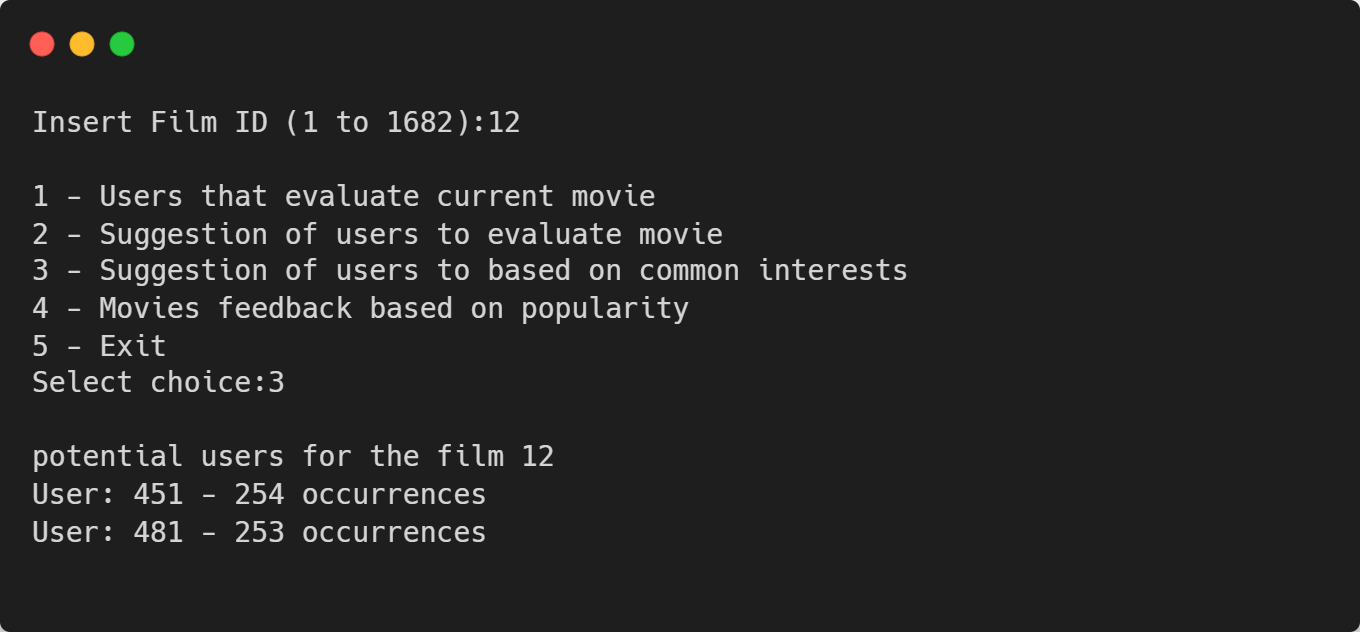


Figura 3 - Output opção 3 para o filme 12

Texto

Descrição gerada automaticamente

Figura 4 - Output da opção 4 para o filme 12

# Conclusão

Podemos concluir que ao fim deste trabalho ficámos a entender melhor sobre o funcionamento das ferramentas MinHash, Funções de Dispersão, Bloom Filter, entre outras. Assim como o funcionamento do MATLAB.

Concluímos ainda que a implementação destas ferramentas facilitam bastante a pesquisa pelas estruturas de dados, evitando assim certos problemas.

# Fontes

- Slides e código das aulas da cadeira de Métodos Probabilísticos para a Engenharia Informática;

- pt.wikipedia.org

- https://www.mathworks.com/help/matlab