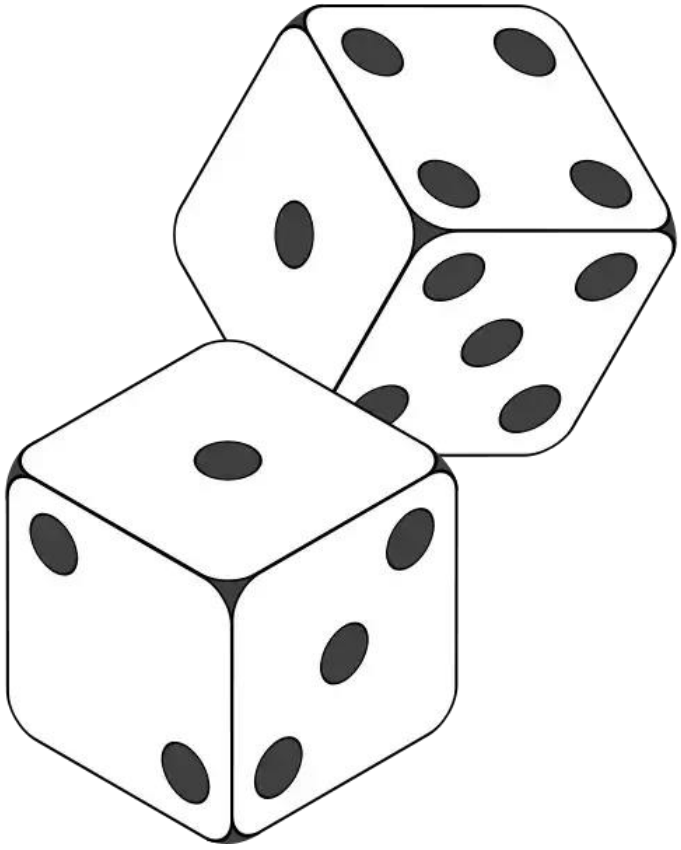
Métodos Probabilísticos para Engenharia Informática



Sistema de disponibilização, procura e sugestão de filmes em Matlab

universidade de aveiro

Realizado por:

- Pedro Ramos n.º 107348

- Gabriel Teixeira n.º 107876

Índice

Introdução do problema

Em diversas aplicações informáticas do nosso dia-a-dia, são utilizados métodos e algoritmos que alteram os temas e os conteúdos que nos são apresentados, mesmo sem nos apercebermos.

Estas técnicas de seleção de conteúdo são fundamentais a qualquer aplicação que precisa de manter os seus utilizadores interessados no que lhes aparece no ecrã, ou que lhes sugira alguma experiência nova, mas com alta probabilidade de os interessar.

Para isto, são necessários algoritmos complexos que analisam a informação disponível de cada utilizador e comparam-na com a informação do conteúdo a mostrar, garantindo assim que o utilizador raramente veja algo que não lhe interessa.

Estes algoritmos também são normalmente aplicados a pesquisas de dados, sendo possível procurar de forma extremamente eficiente entradas numa base de dados partindo de um pedaço de informação incompleta, como por exemplo, procurar nomes de vários filmes numa base de dados a partir de uma única palavra.

O objetivo deste projeto é criar um conjunto destes algoritmos de forma a podermos implementar todas as opções listadas no plano inicial do trabalho.
Estas opções requerem a criação de:

- Um algoritmo que calcule o valor da semelhança entre dois filmes baseado na lista de utilizadores que já viu cada um deles (opção 2);

- Um algoritmo que calcule o valor da semelhança entre dois utilizadores baseado na lista de interesses que cada um tem (opção 3);

- Um algoritmo que calcule uma matriz de assinaturas para os nomes de cada filme, utilizando o método dos k-shingles, para depois obter a similaridade com uma string inserida pelo utilizador (opção 4);

- Um algoritmo que calcule e guarde num Counting Bloom Filter o número de vezes que cada filme foi avaliado com mais de 3 estrelas (opção 4).

Estruturas de dados utilizadas

MinHash

Para evitar as percas de tempo e o gasto de memória encontrado ao comparar cada uma das entradas nas nossas bases de dados, os dados essenciais de cada entrada são reduzidos a um conjunto de assinaturas, de forma a manterem apenas a informação necessária para realizar as comparações e ainda conseguirem identificar onde está guardada a informação completa caso seja necessário (neste caso utilizamos o índice nos vetores com forma de manter a informação sobre os IDs).

Tomando por exemplo a implementação da opção 2, este algoritmo de redução de informação, chamado de minHash, itera por cada filme e começa por aplicar uma função de dispersão a cada ID dos utilizadores que viram o filme. Após obter todos os valores hash para cada utilizador, o algoritmo guarda o valor mínimo obtido entre todos esses valores numa matriz, sendo que cada linha equivale a um filme e cada coluna equivale a uma função de dispersão.

Após isto o algoritmo repete-se, gerando mais valores hash para os utilizadores do mesmo filme e guardando o mais pequeno. Isto acontece k vezes, sendo k o número de funções de dispersão a utilizar, criado uma matriz semelhante à seguinte:

FilmID	Fun. Hash n.º 1	Fun. Hash n.º 2	...	Fun. Hash n.º k
1	43123	829438	...	54780
2	14364	829438	...	48250
...	...	...	...	...

Logo se, por exemplo, utilizarmos 10 funções de dispersão (hash) para 20 filmes iremos obter uma matriz de tamanho 20x10 contendo 10 mínimos para cada filme.

Para comparar a similaridade de Jaccard entre dois filmes apenas precisamos de contar quantas colunas de ambos são iguais e dividir pelo número de colunas (funções de dispersão). Logo se 4 colunas forem iguais a similaridade de Jaccard entre os dois filmes é 4/10 ≃ 0.4

Shingles

Os shingles são secções ou conjuntos importantes de caracteres normalmente utilizados para comparar strings utilizando o minHash.

Neste projeto apenas serviram para transformar o nome de cada filme num vetor contendo k carateres do nome, ou seja, o filme com o nome “Avatar”, utilizando k = 2 shingles, ficaria [“Av”, “va”, “at”, “ta”, “ar”], ou com k=3 shingles, [“Ava”, “vat”, “ata“, ”tar”].

Desta forma, em vez de compararmos os utilizadores por cada filme como mencionámos anteriormente, podemos comparar os shingles gerados a partir do nome de cada filme para encontrar os filmes com os nomes mais parecidos à string inserida pelo utilizador.

Counting Bloom Filters

Os Filtros de Bloom de Contagem são uma variação dos Filtros de Bloom.

São utilizados não só para verificar a existência de um elemento num conjunto, mas também permitem determinar o número de vezes que esse elemento foi adicionado ao conjunto.

O filtro é uma matriz de e bits de tamanho fixo em que as várias posições (índices) são acedidos através de múltiplas funções hash utilizadas para mapear elementos.

No processo de inserção, os Filtros de Bloom de Contagem calculam vários valores de hash para o elemento em questão e incrementam o valor em cada posição correspondente da matriz de bits em 1.

Quando se deseja verificar a pertinência de um elemento no conjunto, o filtro volta novamente a calcular os valores de hash do elemento e verifica os

valores em cada posição da matriz. Se qualquer um desses valores for 0, o elemento com certeza não está no conjunto. Caso contrário, se todos os valores forem maiores do que 0, é possível que o elemento esteja no conjunto, mas não é certo.

Para obter o número de vezes que um determinado elemento foi inserido, a função de verificação guarda todos os valores de cada posição correspondente aos valores de hash e retorna o valor mais pequeno. Isto acontece pois, no caso de haver colisões, um desses valores irá ser a soma das vezes que o elemento desejado foi inserido mais a soma de um outro elemento que colidiu com esse valor, sendo o resultado imprevisível e errado.

Como estes filtros só incrementam, o resultado mínimo é quase sempre o correto (dado que as opções tomadas pelo programador para o tamanho do array de bits e para o número de funções de dispersão são sensatas).

Implementação das opções e funcionalidades requisitadas

Inicio: Ler o ID do filme inserido pelo utilizador e gerar as estruturas de dados relevantes

Inicialmente, o programa começa por pedir ao utilizador que este insira o ID do filme que pretende analisar e comparar.

Após isto o programa percorre por uma lista de funções de forma a criar:

- Os sets necessários à aplicação dos minHashs e que guardam a informação de cada filme/user;

- O set contendo a divisão de cada nome de filme nos k-shingles mencionados anteriormente;

- A matriz de assinatura correspondente ao conjunto de utilizadores que avaliaram cada filme, que depois é transformada numa matriz de distância de Jaccard entre os filmes baseada nos users que viram cada um;

- A matriz de assinatura correspondente ao conjunto de interesses de cada utilizador, que depois é transformada numa matriz de distância de Jaccard entre os utilizadores baseada nos interesses de cada um;

- A matriz de assinaturas correspondente à análise dos k-shingles de cada nome;

- O Counting Bloom Filtrer que guarda o número de vezes que cada filme foi avaliado com mais de 3 estrelas.

Todas estas estruturas são depois guardadas e utilizadas pelo programa principal.

Ciclo principal

O ciclo principal do programa mostra o menu principal e pede ao utilizador que este selecione uma opção.

Após isto o programa divide-se em diferentes caminhos, usando uma função switch case para continuar para o caminho selecionado.

No fim da execução da operação, o programa volta para o menu num novo ciclo, só saindo se o utilizador assim selecionar.

Ex:

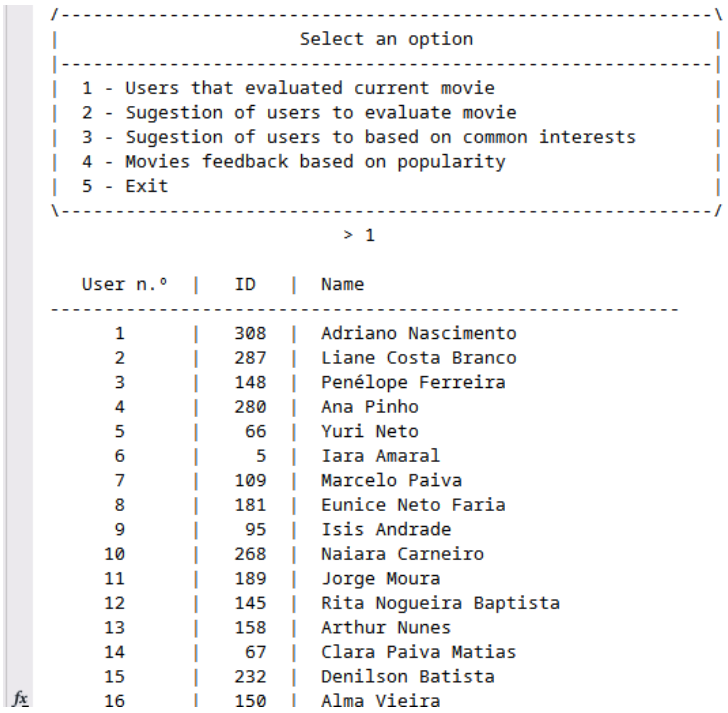
```
>> main
Insert Film ID (1 to 1682): 1
 /------------------------------------------------------------\
 |                      Select an option                      |
 |------------------------------------------------------------|
 |  1 - Users that evaluated current movie                    |
 |  2 - Sugestion of users to evaluate movie                  |
 |  3 - Sugestion of users to based on common interests       |
 |  4 - Movies feedback based on popularity                   |
 |  5 - Exit                                                  |
 \------------------------------------------------------------/
                             >
```

Opção 1: Listar os utilizadores que avaliaram o filme introduzido

Nesta opção o programa simplesmente apresenta uma tabela com as informações (ID e nome completo) de cada utilizador que viu o filme em questão.

A função vai buscar os IDs destes utilizadores ao Set que contem a lista de utilizadores para cada filme, selecionando a linha filmID e iterando sobre os resultados, apresentando-os.

Ex (filmID = 1):



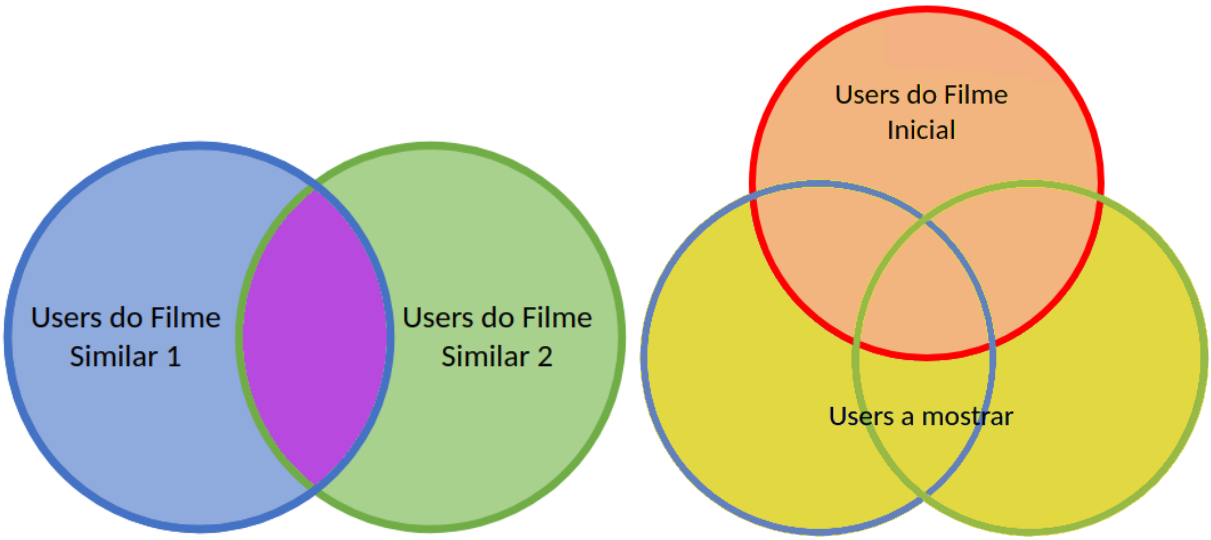
```
/------------------------------------------------------------\
|                      Select an option                      |
|------------------------------------------------------------|
|  1 - Users that evaluated current movie                    |
|  2 - Sugestion of users to evaluate movie                  |
|  3 - Sugestion of users to based on common interests       |
|  4 - Movies feedback based on popularity                   |
|  5 - Exit                                                  |
\------------------------------------------------------------/
                          > 1

   User n.º  |   ID   |  Name
-------------------------------------------------------------
      1      |   308  |  Adriano Nascimento
      2      |   287  |  Liane Costa Branco
      3      |   148  |  Penélope Ferreira
      4      |   280  |  Ana Pinho
      5      |    66  |  Yuri Neto
      6      |     5  |  Iara Amaral
      7      |   109  |  Marcelo Paiva
      8      |   181  |  Eunice Neto Faria
      9      |    95  |  Isis Andrade
     10      |   268  |  Naiara Carneiro
     11      |   189  |  Jorge Moura
     12      |   145  |  Rita Nogueira Baptista
     13      |   158  |  Arthur Nunes
     14      |    67  |  Clara Paiva Matias
     15      |   232  |  Denilson Batista
     16      |   150  |  Alma Vieira
```

Opção 2: Determinar os dois filmes mais semelhantes em termos de utilizadores ao filme original e apresentar os que viram um desses dois filmes, mas não o original

Nesta opção o programa começa por obter os IDs dos filmes mais semelhantes, em termos de utilizadores que os viram, que estão guardados na matriz da distância de Jaccard calculada no início do programa.

Esta matriz encontra-se ordenada previamente por ordem crescente de distância, logo só temos de retirar os IDs da segunda e terceira entrada da matriz. É de notar que a primeira entrada é o ID do próprio filme inicial!

Após obtermos os IDs destes dois filmes, o programa cria um Set com a junção dos IDs dos utilizadores que viram ambos os filmes, a que depois remove os IDs dos utilizadores que já viram o filme inicial:



Ex (filmID = 1):

```
/------------------------------------------------------------\
|                      Select an option                      |
|------------------------------------------------------------|
|  1 - Users that evaluated current movie                    |
|  2 - Sugestion of users to evaluate movie                  |
|  3 - Sugestion of users to based on common interests       |
|  4 - Movies feedback based on popularity                   |
|  5 - Exit                                                  |
\------------------------------------------------------------/
                          > 2

   User n.º  |   ID   |  Name
-------------------------------------------------------------
      1      |     3  |  Adriel Amorim
      2      |     4  |  Ciara Branco
      3      |     7  |  Brenda Carneiro
      4      |     8  |  Abel Domingues Maia
      5      |     9  |  Lia Miranda
      6      |    12  |  Irene Vaz
      7      |    14  |  David Mota Macedo
      8      |    22  |  Noa Cunha Cunha
      9      |    27  |  Kevin Matias Correia
     10      |    28  |  Mateus Fonseca
     11      |    30  |  Nina Amorim Abreu
     12      |    32  |  Alice Antunes
     13      |    37  |  Alice Rocha
```

Opção 3: Encontrar os utilizadores que ainda não viram o filme inicial mas tem gostos muito semelhantes aos que já viram

Nesta opção o programa obtém os IDs dos dois users que seriam os melhores candidatos a ver o filme em questão, baseado nos seus interesses comparados com os dos users que já viram o filme.

Na criação inicial das estruturas, as distâncias de Jaccard entre todos os utilizadores e os que viram o filme inicial, baseadas nos seus gostos, já tinham sido calculadas, gerando uma matriz que compara todos os users aos que já viram o filme. Esta matriz apenas guarda o par de IDs dos users comparados

se a distância de Jaccard entre os utilizadores for menor do que um certo limiar.

Após isto apenas foi calculado quais os dois utilizadores que apareceram mais vezes, usando a função “mode()” do Matlab.

A opção terminal ao apresentar estes dois melhores utilizadores.

Ex (filmID = 1):

```
/-------------------------------------------------------------\
|                      Select an option                       |
|-------------------------------------------------------------|
|  1 - Users that evaluated current movie                     |
|  2 - Sugestion of users to evaluate movie                   |
|  3 - Sugestion of users to based on common interests        |
|  4 - Movies feedback based on popularity                    |
|  5 - Exit                                                   |
\-------------------------------------------------------------/
                           > 3

   User n.º  |   ID   |  Name
-------------------------------------------------------------
      1      |   214  |  Juliana Carvalho Garcia
      2      |   811  |  Martin Neves Figueiredo
```

Opção 4: Procurar os filmes com nome mais semelhante a uma string inserida pelo utilizador

Nesta opção, o programa começa por pedir uma string ao utilizador.

Com esta string, o programa irá gerar os seus k-shingles e irá passá-los por uma função minHash preparada para aceitar valores string.

Nesta função o programa irá gerar as assinaturas da string original, que depois vai comparar com as assinaturas de cada nome de filme, que foram geradas da mesma forma no início do programa.

Esta comparação irá gerar as distâncias de Jaccard entre a string original e os nomes de todos os filmes. Após ordenar a matriz resultante, o programa apresenta os 3 filmes com nomes mais semelhantes à string introduzida.

Ex (string = “story”):

```
/------------------------------------------------------------\
|                      Select an option                      |
|------------------------------------------------------------|
|  1 - Users that evaluated current movie                    |
|  2 - Sugestion of users to evaluate movie                  |
|  3 - Sugestion of users to based on common interests       |
|  4 - Movies feedback based on popularity                   |
|  5 - Exit                                                  |
\------------------------------------------------------------/
                          > 4
Insert a Film Name or part of one: story

  Filme n.º  |   ID   |  Name                        |  N.º Good Reviews
-------------------------------------------------------------------------
      1      |      1 |  Toy Story (1995)            |   417
      2      |   1494 |  Mostro, Il (1994)           |     0
      3      |   1105 |  Firestorm (1998)            |    10
```

O número de boas reviews (mais de 3 estrelas) é obtido ao colocar o ID de cada filme numa função de verificação do Counting Bloom Filter, que retorna o número de boas avaliações do filme inserido.

Este Counting Bloom Filter é preenchido no início do programa ao percorrer o ficheiro “u.data” e, para cada filmID encontrado, incrementar os k valores no Filtro correspondentes aos códigos hash gerados pela função para cada filmID.

O número de boas reviews foi confirmado postreormente ao analisar o ficheiro “u.data” e todas as entradas nos pareceram corretas.

Opções escolhidas na implementação dos métodos probabilísticos

Funções Hash

Talvez uma das partes mais importantes deste projeto foi a seleção das opções para cada estrutura utilizada.

Para as funções de dispersão, foram utilizadas as funções do tipo DJB31MA, visto que estas funções aparentam ter um número ligeiramente menor de colisões e, dado à sua simplicidade, são bastante rápidas de correr.

A "seed" utilizada foi 255, pois foram testadas várias "seeds" mas, desde que os valores não fossem muito pequenos, não afetavam notavelmente a exatidão dos algoritmos.

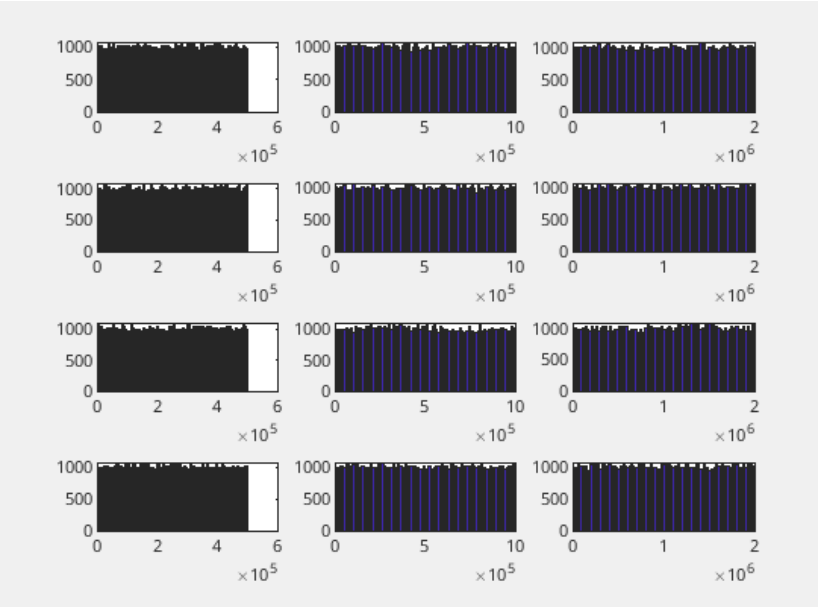
A diferença entre esta função e as funções string2hash djb2 e string2hash sdbm não é grande o suficiente para merecer uma comparação, mas é de notar que a função hashstring é bastante pior em termos de colisões, como evidenciado na seguinte tabela:

```
numColisoes =

        8729        4605        2407
        8858        4722        2455
       12405        9062        6560
        8774        4607        2382
```

Descrição: Números de colisões obtidos, cada linha representa uma função (1: string2hash djb2, 2: string2hash sdbm, 3:hashstring, 4:DJB31MA) e cada coluna representa o número de valores hash disponíveis (1: 5x10^5, 2:1x10^6, 3: 2x10^6)

A comparação entre a distribuição de valores hash também pode ser vista no próximo gráfico, em que notamos que a função hashstring (terceira linha) tem uma distribuição mais desigualada independentemente do número de valores hash disponíveis. A função DJB31MA foi também testada com várias "seeds"

diferentes e, desde que não sejam menores que ~50, não afetavam muito os valores (aqui foi utilizado seed = 255).



Descrição: Histogramas dos valores de hash obtidos, cada linha representa uma função (1: string2hash djb2, 2: string2hash sdbm, 3:hashstring, 4:DJB31MA) e cada coluna representa o número de valores hash disponíveis (1: 5x10^5, 2:1x10^6, 3: 2x10^6)

O número de funções hash utilizado para a geração das matrizes de assinaturas correspondentes aos utilizadores por filme, aos interesses por utilizador e nos k-shingles dos nomes dos filmes foram 500. Este número foi escolhido pois não é grande o suficiente para fazer o utilizador ter de esperar mais de 10 segundos para o código inicial correr mas é grande o suficiente para que os resultados finais tenham um elevado nível de precisão (Distâncias de Jaccard muito parecidas aos valores teóricos).

Counting Bloom Filter

O nosso Filtro de Bloom foi inicializado com um tamanho de 10000, ou seja, as funções de dispersão utilizadas no filtro têm 10000 valores de hash possíveis. Este número foi escolhido pois existem um pouco menos de 2000 filmes, e como ire-mos utilizar 5 funções de dispersão por filme, obtemos 10000 valores de hash (teoricamente, pois existem sempre colisões).

Como mencionado, para a função insert() e verify() do Counting Bloom Filter foram utilizadas 5 funções de hash, pois neste caso, como temos um número limitado de valores de hash disponíveis, temos de escolher um número pequeno o suficiente para que não existam imensas colisões mas grande o suficiente para que, caso alguma aconteça, não exista corrupção dos resultados, visto que o Counting Bloom Filter é capaz de tratar de colisões, pois fica sempre com o menor de todos os encontrados.

Shingles

Como mencionado no início do projeto, os shingles dividem as strings em conjuntos de carateres seguidos.

Neste projeto foi-nos aconselhado utilizar shingles de tamanho 2 a 5.

Na nossa implementação, shingles de tamanhos pequenos levam a que as palavras mais semelhantes à string procurada não contenha exatamente as palavras procuradas, mas shingles de tamanho maior encontravam muito poucas semelhanças, por exemplo, procurar por “story” leva a muitos filmes contendo shingles como “st”, “or” e “ry” (k = 2) mas existem muitos poucos filmes que contem o shingle “stor” ou “tory” (k=4).

Concluindo, escolhemos utilizar shingles de tamanho 3 visto que assim recebemos resultados semelhantes à string procurada e ainda garantimos que existem bastantes filmes com shingles parecidos.

Conclusão

Dadas as entradas inseridas e uma análise dos ficheiros recebidos, os resultados finais parecem ser corretos.

As opções 1 e 4 podem ser, pela maior parte, confirmadas como corretas ao analisar os ficheiros mencionados.

As opções 2 e 3 são mais difíceis de confirmar, visto que dependem de muitos fatores como o número de funções de dispersão utilizadas, mas ao verificar alguns resultados obtidos pelo programa e ao compará-los com os valores dos ficheiros dados (por exemplo, se o programa retorna dois users parecidos com IDs 1 e 5, podemos ir ao ficheiro que guarda os seus interesses e confirmar que de facto estes users tem vários interesses em comum).

Ao longo deste projeto consolidamos muitos dos assuntos abordados durante as aulas práticas e teóricas, especialmente Filtros de Bloom de Contagem, Shingles e MinHash.

Foi uma oportunidade valiosa para adquirir conhecimentos e habilidades importantes no âmbito da manipulação de grandes conjuntos de dados e da criação de algoritmos para a descoberta da semelhança de conjuntos, o que nos permite aplicar estes tópicos em desafios do mundo atual e futuro.

Acreditamos que esta aprendizagem nos ajudará a desenvolver a nossa carreira e agradecemos a todos os profissionais e instituições envolvidos na realização destas aulas, pois sem o seu apoio e dedicação, não teríamos sido capazes de adquirir o conhecimento necessário para a realização deste projeto.