Hábitos de Compra
Relatório

Departamento de eletrónica, telecomunicações e informática

Universidade de Aveiro

João Diogo Videira Oliveira

(93295) jdvoliveira@ua.pt

Gabriel Filipe Teixeira Ribeiro

(93036) gabrielribeiro00@ua.pt

Dezembro de 2019

Introdução

No âmbito da unidade curricular Métodos Probabilísticos para Engenharia Informática, foi-nos proposto a realização de um trabalho prático em Java para a consolidação dos conceitos abordados nas aulas teóricas e práticas.

O trabalho tem como objetivo desenvolver uma aplicação que demonstre a utilização conjunta de deteção de elementos similares e de deteção de pertença a um conjunto.

O trabalho escolhido para o nosso grupo foi hábitos de compras, no qual desenvolvemos módulos como Bloom Filter e MinHash.

Este documento encontra-se dividido por partes. Após esta introdução, serão apresentados os testes dos 2 módulos mais importantes para o correto funcionamento deste trabalho prático, são eles Bloom Filter e MinHash. Por fim será exposta uma conclusão do trabalho na qual é identificado o objetivo inicial bem como o cumprimento do mesmo.

Manual de utilizador

Este projeto possui um menu (Figura 1) onde é possível adicionar clientes individualmente, assim como, gerar 5 de uma só vez sem repetição em ambos os casos, podemos também obter uma lista de todos os clientes adicionados e uma lista de compras de cada cliente, é ainda possível verificar a similaridade nas compras efetuadas por cada cliente. Este menu encontra-se quando o ficheiro “main.java” é executado.

```
Menu
1: Adicionar cliente
2: Mostrar lista de clientes
3: Mostrar lista de compras de um cliente
4: Gerar 5 Clientes
5: Encontrar Semelhantes
6: Numero de clientes na base de dados
0: Terminar programa
Opção:
```

Figura 1 – Menu

Teste Módulo Bloom Filter

Para testar este módulo é necessário inserir elementos no Bloom Filter, para que este faça a respetiva verificação da existência dos mesmos, ou seja, verifica se o elemento já foi ou não inserido anteriormente. Neste Bloom Filter foi usada uma fórmula para definir o tamanho do filtro. Esta fórmula usa uma percentagem aceitável de falsos positivos.

Após essa verificação existem 2 casos possíveis:

1. O elemento já existe e, portanto, não é inserido.
   Neste caso é impressa uma mensagem no terminal do tipo “nome já está na lista”.

2. O elemento não existe e é inserido.
   Neste caso o elemento é inserido sem qualquer mensagem no terminal para que não seja impressa demasiada informação no mesmo e, assim, facilitar a sua leitura.

É apresentado um teste com nomes a serem inseridos, onde é exposta informação acerca dos falsos positivos, incluindo a sua probabilidade. (ver segundo parágrafo apresentado na Figura 2).

Para além dos falsos positivos também apresenta informação acerca do próprio Bloom Filter, mais precisamente o tamanho do filtro, o número de elementos e o número de Hash Functions (ver terceiro parágrafo apresentado na Figura 2).

Para além do teste com nomes, é também apresentado um teste com strings aleatórias a serem inseridas, no qual é exposta a mesma informação referida anteriormente (ver dois últimos parágrafos apresentado na Figura 2).

```
Gabriel Ribeiro já está na lista

(Teste com os nomes)Probabilidade de falsos positivos  1,1400% :
A probabilidade de falsos positivos foi definida quando criamos o bloom filter.
Esta é 0.1 que corresponde ao valor apresentado

Bloom Filter
Tamanho do filtro  =  95851;
Número Elementos  =  10000;
Número de Hash Functions  =  7

(Teste com strings random)Probabilidade de falsos positivos  0,9950% :
A probabilidade de falsos positivos foi definida quando criamos o bloom filter.
Esta é 0.1 que corresponde ao valor apresentado

Bloom Filter
Tamanho do filtro  =  958506;
Número Elementos  =  100000;
Número de Hash Functions  =  7
```

Figura 2 – Terminal do teste do Bloom Filter

Teste Módulo MinHash

A MinHash é utilizada para verificar a similaridade entre elementos. Neste caso, este módulo vai servir para a verificação da similaridade das listas de compras dos diferentes clientes.

Para testar este módulo, primeiro é apresentado um teste com frases predefinidas, onde se pode verificar a percentagem de similaridade entre essas frases (ver primeira parte da Figura 3).

De seguida é apresentado um outro teste com strings aleatórias, no qual se verifica o número de strings que são 10% semelhantes. Este último pode demorar um pouco de tempo a ser realizado devido ao número elevado de strings (ver segunda parte da Figura 3).

```
Teste com frases predefinidas
[a, garrafa, rato, roeu]
[a, bateu, cabeça, com, girafa]
[a, garrafa, roeu]
[a, garrafa, rato, roeu]
Similariedade entre "rato roeu a garrafa" e "girafa bateu com a cabeça": 3,09%
Similariedade entre "rato roeu a garrafa" e "rato roeu a garrafa": 100,00%
Similariedade entre "rato roeu a garrafa" e "roeu a garrafa": 81,82%
Similariedade entre str5 e str6: 19,05%

Teste com strings random:
Loading...
Done!Número de strings que sao 10.0% semelhantes: 948
```

Figura 3 – Terminal do teste do Bloom Filter

Após efetuados testes, verificámos que a MinHash funciona substancialmente melhor com 50 hash functions do que com 100. Verificámos também que com a utilização no MinHash da hash function usada no Bloom Filter, não correu como esperado, por isso usamos a hashcode do java no MinHash.

Conclusão

Através deste trabalho prático, desenvolvemos os nossos conhecimentos principalmente acerca dos módulos Bloom Filter e MinHash, devido ao desenvolvimento e implementação do código destes módulos em Java.

O nosso objetivo era desenvolver uma aplicação eficaz que realizasse todas as funções corretamente com a implementação dos módulos referidos anteriormente.

O projeto sofreu algumas alterações relativamente ao enviado nas 2 submissões intermédias, visto que surgiram alguns obstáculos, mas de uma forma geral o objetivo foi alcançado e a utilização dos conteúdos abordados nas aulas teóricas e práticas foi indispensável para uma boa abordagem e realização deste trabalho prático.