Engenharia de Computadores e Telemática
Métodos Probabilísticos para engenharia informática
Universidade de Aveiro

Projeto 8

Processamento de reviews de jogos

Turma Prática 5,
Autores:

- Guilherme Amaral Ribeiro Pereira: 93134
- José Luís Rodrigues Costa: 92996

2019/2020

Índice

INFORMAÇÕES E FORMATO DO DATASET

Para o teste de todo o projeto desenvolvido, tanto nos testes como na aplicação conjunta, é fornecido 3 datasets de diferentes tamanhos. Em todos estes utilizamos informação recolhida de 2 websites (Steam e GOG).

O dataset com nome " BigData.csv " tem um tamanho de aproximadamente 300'000 linhas que contém as informações apresentadas na imagem nesta página. O dataset com nome "MedData.csv" contém cerca de 150'000 linhas e o "SmallData.csv" contém cerca de 50'000 linhas.

Para obter valores corretos é necessário ter em conta a formatação dos dados a inserir na aplicação conjunta bem como nos testes que necessitarem desses dados.

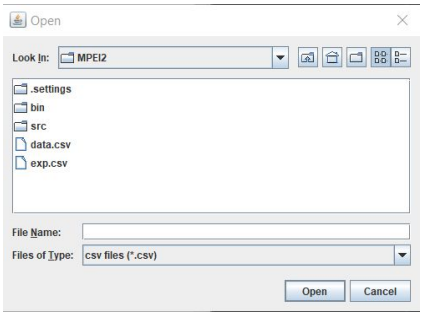
O formato é um ficheiro .csv onde se deverá colocar os dados , este deverá ter em cada linha o utilizador, seguido de recomendações, reviews e titulo do jogo, todos estes valores separados por uma vírgula (utilizador,recomendação,review,título). Para o campo recomendações só é válido o valor Recommended ou Not Recommended.

utilizador	recommendation	review	title
194234494	Recommended	This game is fun challengi	Rocket League®
485063485	Recommended	Brilliant game still in alpha	Rust

COUNTING BLOOM FILTER TESTES

Para testar o funcionamento do Counting Bloom Filter foram criados três testes nos quais verificamos o seu correto funcionamento para um conjunto de elevada dimensão, pequena dimensão e o conjunto de dados a ser utilizado na aplicação principal.

(Para correr os testes BigTest.java e SmallTest.java basta correr o respetivo .java no terminal e os resultados são apresentados sem necessidade de mais nenhum input,para o TestWithDataSet.java além de o correr no terminal será pedido para selecionar o ficheiro .csv que tenha os dados a serem utilizados na aplicação conjunta)



Os testes focaram-se em:

- Verificar para diferentes valores k(número de hash Functions) quais geram a menor probabilidade de falsos positivos, incluindo o k ótimo calculado e usado pelo Counting Bloom Filter.
- Comparar o valor da probabilidade de falsos positivos teórica com a probabilidade de falsos positivos existentes no Counting Bloom Filter após serem inseridos valores.
- Verificar se o número de frases iguais inseridas no Counting Bloom Filter é igual ao valor contado pelo mesmo.
- Testar o número de colisões

ANÁLISE DOS RESULTADOS OBTIDOS NOS TESTES AO COUNTING BLOOM FILTER

Para testar se a probabilidade de falsos positivos no Counting Bloom Filter era igual á probabilidade teórica, no teste BigTest.java foram geradas Strings aleatoriamente e inseridas no Counting Bloom Filter com um vetor interno 8x maior que as strings inseridas, de seguida geramos novas Strings diferentes e usando a função isMember() que verifica se nenhuma das posições geradas pelas hash functions está a zero,verificamos quantas vezes a função retorna verdade, e em seguida dividiu-se esse número pelo número de Strings que entraram no método isMember().

```
Inicio do BigTest para o BloomFilter
Numero de linhas aleatorias a inserir: 6686
Inserindo no fitro:
**********
Concluido com sucesso
-> Melhor numero de HashFunctions aplicadas (k): 6
-> Provabilidade de falsos positivos (Teorica) : 0,02158
-> Tamanho interno do vetor: 53488
Probabilidade real de falsos positivos no Bloom Filter: 0,06626
------------------------------------------------------------
Probabilidade real de falsos positivos utilizando diferentes k's :

K->2,P->0,09064
K->6,P->0,07299
K->9,P->0,08047
K->20,P->0,08794

------------------------------
Testar o contador do BloomFilter
Inserindo respetivamente, 85x, 147x, 302x, 597x, 467x, 5 Strings Diferentes

String 0->peaK
String 1->hnSm
String 2->mf3m
String 3->gDIQ
String 4->Fy9C

Contagem da String 0->85
Contagem da String 1->147
Contagem da String 2->302
Contagem da String 3->597
Contagem da String 4->467
```

BigTest.java

```
Wings of Prey: Special Edition          ->      8       |       8       igualdade?: true
Deponia 2: Chaos on Deponia             ->      24      |       24      igualdade?: true
Niche - a genetics survival game        ->      7       |       7       igualdade?: true
Gothic                                  ->      99      |       100     igualdade?: false
Empires of the Undergrowth              ->      3       |       3       igualdade?: true
Pillars of Eternity: The White March - Part II  ->      3       |       3       igualdade?: true
Day of the Tentacle Remastered          ->      53      |       53      igualdade?: true
The Park                                ->      10      |       10      igualdade?: true
Fallout 3: Game of the Year Edition     ->      127     |       127     igualdade?: true
Jade Empire: Special Edition            ->      117     |       117     igualdade?: true
Bard's Tale  The                        ->      49      |       49      igualdade?: true
------------------------------------------------------------------------------------------------------------------------------------------
Jogos                                                   Contagem Ficheiro       Contagem BloomFilter
------------------------------------------------------------------------------------------------------------------------------------------
Fim do Test com valores do banco de dados usado na aplicacao conjunta para o BloomFilter
-> Numero de reviews por jogo no banco de dados e no BloomFilter:
==========================================================================================================================================
Percentagem de erro: 7,68895%
```

TestWithDataSet.java

Para testar se o K(número de Hash Functions) gerado pelo Counting Bloom Filter era o melhor criamos vários Objetos Bloom Filter com vários valores de K e aplicamos o mesmo processo de cálculo de falsos positivos descrito em cima a todos de modo a poder comparar o valor das probabilidades.Como se pode ver na imagem e no teste se reparamos que um dos K's gerados é de facto o que nos dá uma probabilidade de falsos positivos mais baixa.O que vai de acordo com fórmula teórica(K=6 neste caso).

Para testar se o valor de String iguais inseridas no Counting Bloom Filter é igual ao valor retornado pelo mesmo a quando da procura das mesmas introduzimos no Counting Bloom Filter várias Strings aleatórias um número aleatório de vezes, e verificamos que a contagem é igual ao número de inserções de cada.

Para o teste em que utilizamos o dataset do projeto (TestWithDataSet.java), verificamos que para todos os jogos, o número de contagens real é igual ou muito próximo do valor fornecido pelo Counting Bloom Filter, apresentando também no fim a percentagem de erro do mesmo.

MINHASH

Para testar o funcionamento do módulo MinHash foi criado o teste BigTest.java onde verificamos o seu funcionamento perante dados relativos à aplicação.

Este BigTest.java apresenta uma comparação pequena de 10 reviews na consola em que utilizamos o algoritmo MinHash desenvolvido para obter a similaridade experimental (Jaccard) (em que apenas calculamos a matriz de assinaturas correspondente as reviews a comprar, logo não é calculada a matriz de assinaturas total do .csv), como também o cálculo da similaridade real das mesmas, utilizando a similaridade de Jaccard e considerando similaridade acima de 80% em ambos os cálculos.

Depois da apresentação destas informações, é calculado o erro provocado pelo algoritmo do MinHash utilizando 100 Hash Functions e comparando 50000 Strings aleatórias do documento, que é apresentado no fim da imagem (demora cerca de 30s a fazer estas comparações).

(Para correr o BigTest.java além de o correr no terminal será pedido para selecionar o ficheiro .csv que tenha os dados a serem utilizados na aplicação conjunta conforme a imagem apresentada na página 3).

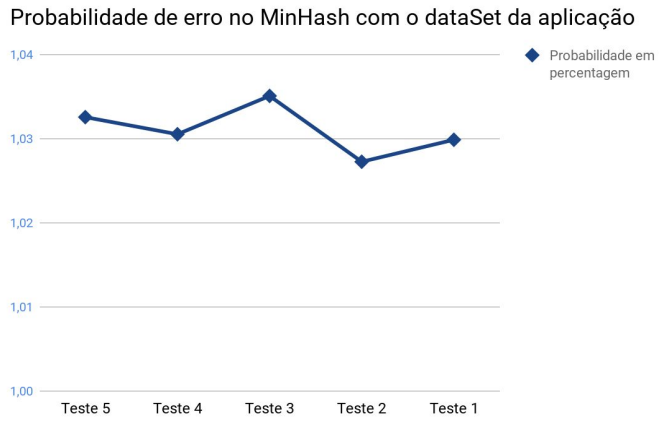
Na imagem apresentada, que é um teste utilizando o BigData.csv, o erro para todas as Strings comparadas é muito pequeno, e por tanto uma boa aproximação ao valor real da similaridade (Jaccard).

No gráfico apresentado em baixo, após vários testes ao BigTest.java, concluímos que a percentagem de erro é sempre muito perto de 1%.

```
Teste ao MinHash:
Para este teste vamos comparar 10 reviews aleatórias do documento selecionado.
Strings a comprar:

0- Easy to learm game but hard to master. You wil lline up the perfect hit only to completely miss and score a goal you have no idea how i went it.
Distancia exprimental (minHash): 0.0
Distancia teorica (real): 0.0
----------------------------
2- GTA Online is not worth playing anymore unless you have tons of money burning a hole in your pocket.Here are a few points Prices for DLC items ar
Distancia exprimental (minHash): 0.01
Distancia teorica (real): 0.01096774193548387
----------------------------
4- this game 4 hours of gameplay got ban 30 days for nothing i was still trying to figure out how the game was working i did not use any hack or moc
Distancia exprimental (minHash): 0.02
Distancia teorica (real): 0.018604651162790697
----------------------------
6- you may think it is good but it isnt idk if this is just me but my head will not move ive done nuthing to make it do this.ive reported it about 5
Distancia exprimental (minHash): 0.03
Distancia teorica (real): 0.02631578947368421
----------------------------
8- dis gam gud. I lyk munstre hontar arlot. I pleyed at list too hadred ours ov it.10/10  ->  9-"Game doesnt even work anymore in early acess i coul
Distancia exprimental (minHash): 0.01
Distancia teorica (real): 0.0
----------------------------

Erro médio para a totalidade do ficheiro aplicando MinHash (Erro no calculo da similaridade de 50000 Strings pelo MinHash)
***********
1,02987%
```



APLICAÇÃO CONJUNTA

A aplicação conjunta consistem em juntar os módulos criados(MinHash e Counting BloomFilter) e através destes conseguir executar uma série de operações sobre os dados de um ficheiro .csv.
Para executar este programa primeiro deverá executá-lo no terminal,em seguida será pedido que insira um valor para o número de Hash Functions a serem usadas no MinHash (de 0 a 100) sendo 50 o valor recomendado, visto que o número de hash functions escolhidas terá impacto no tempo necessário para gerar a matriz de assinaturas.Após gerada aparecerá no terminal um menu com as várias operações que a aplicação realiza. Nesta aplicação o minhash apenas é utilizado de maneiras diferentes sobre as reviews dos utilizadores.

```
Selecione o valor de hashFunctions a utilizar no minHash (1<k<100):

BigData - Tempo médio de espera para k=50 -> 55s
MedData - Tempo médio de espera para k=50 -> 25s
SmallData - Tempo médio de espera para k=50 -> 8s
50
Criando Matriz de Assinaturas:
*****
Métodos Probabilisticos para Engenharia Informática
Projeto 8
Aplicação Conjunta:
Menu:
0 -> Sair da Aplicação;
1 -> Verificar se um jogo tem reviews
2 -> Mostrar o numero reviews de um jogo
3 -> Coleção de reviews de um jogo (Sem reviews muito parecidas)
4 -> Percentagem de reviews iguais de 2 utilizadores
5 -> Percentagem por jogo de recomendações positias
```

Opção 1 -> Após selecionar esta opção será pedido para introduzir um nome de um jogo para o qual , utilizando o Counting BloomFilter, poderá verificar se o jogo inserido contém reviews.
Exemplo: Grand Theft Auto V , UFO: Aftermath

```
Proximo comando: 1

Nome do jogo : Infinium Strike Demo
tem reviews? = true


Proximo comando:
```

Opção 2 -> Após selecionar esta opção será pedido para introduzir um nome de um jogo para o qual , utilizando o Counting BloomFilter, poderá verificar a quantidade de reviews que jogo inserido contém.
Exemplo: Infinium Strike Demo , Grand Theft Auto V

```
Proximo comando: 2

Nome do jogo : Rocket League®
numero de reviews = 39998
```

Opção 3 -> Após selecionar esta opção será pedido para introduzir um nome de um jogo para o qual , utilizando o MinHash,é gerado uma lista de reviews sem comentários parecidos, ou seja, comentários que tenham uma similaridade menor a 80%, visto que não tem interesse a leitura de comentários muito similares. Após a lista ser gerada, é apresentado na consola 5 reviews (pouco similares) e as restantes são guardadas num ficheiro com o seguinte formato : "Reviews(nomeDoJogo).txt", que apresenta as reviews e os utilizadores que as escreveram.
Para além destas funcionalidades, também apresenta o número de reviews similares detectado e o número de comparações de reviews efetuadas para o jogo selecionado.
Esta operação pode demorar cerca de 30s, dependente da quantidade de reviews que o jogo tem no ficheiro .csv, sendo um perfeito exemplo disto, os exemplos apresentados em baixo.
Exemplo: Dead by Daylight , Inked

```
Proximo comando: 3

Nome do jogo : Dead by Daylight

Como o numero de reviews pode ser extenso, apenas vamos aprensentar na consola 5 exemplos, sendo todos os ou

***
Review 0-> "Recommended for it's potential to grow as developers are super active (weekly live streams Q&amp

Review 1-> hehe are funi to infinites juke spots heheEspacally the tree with henging mans and the skool buss

Review 2-> Amazing Surival Horror game!I love playing as the Killer and Survivor!The killers are all awesome

Review 3-> i got this for free from a friend hence the number of hours i've put into it. the game can be fun

Review 4-> i have fun with this game. BUT the new ranking system sucks but if u dont care about rank buy it!

Numero de reviews similares detectado: 1204
Comparações efetuadas: 52033750
```

Opção 4 -> Após selecionar esta opção será pedido para introduzir o id presente no ficheiro .csv de 2 utilizadores que, utilizando o MinHash, verifica a percentagem de reviews similares entre os utilizadores.
Exemplo de id: 172356913 <-> 244270722

```
Proximo comando: 4

Qual os ID's dos utilizadores que pretende comparar?
Utilizador 1-
516198877
Utilizador 2-
179483582

Quantidade de reviews de 516198877: 4
Quantidade de reviews de 179483582: 4

Percentagem de similaridade de reviews: 12,5%
```

Opção 5 -> Após selecionar esta opção através do Counting BloomFilter , será escrito para um ficheiro com o nome ReviewsPercentagem.txt todos os jogos e a respetiva percentagem de recomendações pelos utilizadores, e no terminal 5 exemplos.

```
Proximo comando: 5

Como o numero de jogos pode ser extenso, apenas vamos aprensentar na consola 5 exemplos, sendo todos os outros guardados num ficheiro com nome ReviewsPercentagem.txt

Factorio=100.0%
PLAYERUNKNOWN'S BATTLEGROUNDS=52.372539542325704%
Wallpaper Engine=100.0%
Slay the Spire=99.23076923076923%
RimWorld=100.0%
```