Projeto VIDEOCLUB – Relatório MPEI

Mariana Pinto (84792 - mariana17@ua.pt)
Tomás Freitas (84957 - tomasfreitas@ua.pt)

Data de Entrega: 03/12/2017

1 Introdução

Neste relatório será apresentada a metodologia exercida para a criação de uma aplicação tipo "VIDEOCLUBE", com o objetivo de usar os métodos lecionados durante o semestre em MPEI. Fazendo uso da linguagem JAVA, é criada uma ferramenta em que o utilizador interage com o programa através do método "ManagmentMenu", com um menu (numerado de 1 a 10) que permite ao cliente:

- Adicionar um user;
- Remover um user;
- Procurar um filme;
- Listar Filmes;
- Recomendar um filme baseado em atores;
- Recomendar um filme baseado noutro filme;
- Devolver um filme;
- Requesitar um filme;
- Ver os users que requesitaram um dado filme;
- Listar os filmes por Rating;

Os módulos utilizados (para além dos já enunciados) são "ActorSimilarity", "BloomFilter", "LSHSimilarity", "MinHash" e "ShingleCreate". Cada um deles será explicado posteriormente, bem como o funcionamento de todas as funcionalidades existentes.

2 Modules

2.1 Similarity

Este módulo usa duas strings para verificar a sua similariedade. Dada a string um e a string dois, são então criadas as shingle para as respetivas frases.

O número único de minhash é, então, calculado apartir da shingle criada de cada string, de forma a que as funções "getcommon" e "getunique" possam calcular os parâmetros comuns e diferentes das duas minhash calculadas. É então dividido o numero de minhash comuns pelo número de minhash diferentes, resultando no indice de similariedade (semelhança).

A partir dái, conseguiamos obter a distância de jaccard fazendo 1-semelhança.

2.1.1 ActorSimilarity

O módulo *"ActorSimilarity"* é um módulo específico do "Similarity". Enquanto que o segundo nao é usado no algoritmo (é apresentado por ser uma versão genérica do que queremos realmente alcançar), o primeiro usa a classe "Movie", presente no "VideoClube", e é usado no trabalho.

Este módulo procura no dataset os filmes em que o ator participou, verificando quais os filmes que apresentam no elenco o mesmo nome que o ator pretendido e devolve uma lista com essas mesmas participações.

2.2 BloomFilter

Implemetou-se um BloomFilter, em JAVA, que fosse paramatrizável tanto em termos de tamanho de dados como no número de hashfunctions utilizadas. As seeds criadas são as mesmas utilizadas em cada instância do programa pois o array *seeds* é estático e final.

A função "generateSeeds" é responsável por gerar tantos números aleatórios, quanto as hash functions para que tenhamos hashfunctions distintas.

Enquanto que a função "add" adiciona uma string ao BloomFilter, a função "hash" cria as hash para cada string que se quer carregar e a "isMember" verifica se uma string é um possível membro do BloomFilter.

2.3 LSHSimilarity

Este módulo, apesar de não estar 100% concluído, atualmente cria bandas utilizando os filmes no nosso dataset. Dizemos que não está a 100% porque, apesar de nos dar colisões bastante fiáveis (diga-se nas sagas tipo Harry Potter ou Lord of the Rings), também retornava colisões bastante díspares.

Para não perder o rendimento esperado na procura decidimos não o usar no trabalho, pois ainda só tivemos oportunidade de abordar estes temas nas aulas teóricas e tornava-se mais complicado descobrir a origem do erro.

O objetivo era comparar filmes que tivessem uma distancia de jaccard similar, verificando se estes eram relacionados ou não (exemplos de sagas, trilogias ou remakes de filmes) através do seu nome.

2.4 MinHash

O módulo MinHash cria o número de MinHashes (mais uma vez, parametrizável) associadas a um set de shingles.

A função "minHasher" cria as minhash para cada shingle presente no Set e a função "seedgenerator" tem a mesma funcionalidade que a função "generateSeeds" usada no módulo BloomFilter.

2.5 ShingleCreate

O módulo ShingleCreate cria os Shingles únicos associados a uma certa String. No nosso caso, a String utilizada representa o elenco de um filme.

A função CreateShingle usa uma string para criar shingles únicas, usadas depois ao longo do trabalho para vários fins (como para as MinHash, já explicadas anteriormente).

3 VideoClube

O "ManagmentMenu" é a prncipal ferramenta de interação entre o utilizador e o programa construído. Através dele é possível, através de um menu interativo:

- 1- Adicionar um user, através da função "adduser";
- 2- Remover um user, através da função "removeUser";
- 3- Procurar um filme, através da função "searchforamovie";
- 4- Listar Filmes, através da função "listMovies";
- 5- Recomendar um filme baseado em atores, através da função "RecommendMovie";
- 6- Recomendar um filme baseado noutro filme, através da função "recommendmovie" (notar a diferença nas maiúsculas desta função para a última citada);
- 7- Devolver um filme, através da função "checkout";
- 8- Requesitar um filme, através da função "checkin";
- 9- Ver os users que requesitaram um dado filme, através da função "listcheckoutuser";
- 10- Listar os filmes por Rating, através da função "listMoviesByRating";

Todos os filmes são carregados num BloomFilter de forma a que se consiga procurar o filme pretendido, tal como está implementado na função "createbloom". A função "readMovies" lê os filmes existentes no dataset do ficheiro "finaldata.csv" presente na source do código, a "Nametoid" associa a cada filme o seu ID e a "clientid" retorna o ID de um dado cliente.

Este método combina todos os outros existentes na pasta VideoClube, direta ou indiretamente. Nesta pasta estão implementados os ficheiros referentes à manipulação dos dados do dataset, como os filmes, users e outros dataset guardados e necessários para correr o videoclube.

No ficheiro "Client" é possível obter os dados do cliente registado no videoclube, como o nome, Cartão de Cidadão e o ID de Cliente associado à sua conta. O ficheiro "Data" é usado para gerar a data de inscrição de um user aquando da funcionalidade número um do "ManagmenteMenu", "Movie" para recuperar todos os dados existentes sobre determinado filme presente no dataset, enquanto que o "MovieDatabase" é o responsável pelas alterações no dataset, por exemplo as funcionalidades 7 e 8 do menu interativo. "UserManagment" trata das funcionalidades como adicionar e remover users e, finalmente, "Student" é uma classe derivada de cliente, onde os dados trabalhados são o nome do estudante, número mecanográfico ou curso.

4 TestingArea

Nesta área foram realizados testes maioritariamente em linguagem JAVA (apenas 3 testes em MATLAB, apesar de todos os valores obtidos nos testes em JAVA serem compatívéis com os valores teóricos) aos módulos usados no trabalho.

4.1 TestBloomFilter

No teste ao BloomFilter são, essencialmente, criados dois arrays na qual o array2 tem, em posições random, os elementos do array original. Inserimos o array 2 no bloomfilter e tentamos encontrar os elementos que sabemos que estão lá, sendo esses os elementos do array original.

O BloomFilter tem de tamanho 1000000 e 10 hashfunctions (como referimos anteriormente, ambas as variáveis são parametrizáveis), o número de elementos é um valor muito baixo para o valor total do bloomfilter. A percentagem de existirem falsos positivos é demasiado baixa, por isso é-nos (quase) sempre calculado que existem zero falsos positivos.

Foi gerado um código em MATLAB, presente na pasta "MATLAB" dentro da pasta "TestingArea", de forma a criar um BloomFilter com duas strings aleatórias e comparar as mesmas verificando quais deles são prováveis de existirem no BloomFilter. Os resultados são compatíveis com os obtidos anteriormente.

4.2 TestActorSimilarity

Neste teste, é usado o nome de um dado ator para procurar todos os filmes presentes no dataset em que este esteve presente no elenco. Trata-se de um teste ao módulo ActorSimilarity que, dado o exemplo do ator Leonardo DiCaprio, apresenta uma lista com os nomes e o elenco com os restantes atores em que

este teve presente. O teste foi corrido com sucesso e apresentou os resultados esperados.

4.3 TestHashing

Com a "TestHashing" é testado o algoritmo para criação da hash através de strings. Com a str "olá" é gerada o valor da hash correspondente, usando os módulos do package "murmur", devidamente credenciados (open-source code) no início de cada código.

O valor teórico pode ser confirmado pelo teste presente na pasta "MATLAB" dentro da pasta "TestingArea", através do algoritmo gerado em MATLAB pelo ficheiro "testes".

4.4 TestShingle

Realizando o teste ao método ShingleCreate, é usado um set de string a partir da string "Penso, logo existo". É retornada então todas as shingles possíveis de gerar com a dada frase.

4.5 TestMinHasher

Este teste verifica o módulo "MinHash" já explicado. Dada a string "Penso, logo existo", é usado o set de shingles, gerado a partir do set de strings da frase "Penso, logo existo" (usando a função "createShingle" presente em ShingleCreate), para produzir o número de minhash associado ao set de shingles, de forma a que cada shingle tenha uma minhash única.

4.6 TestSimilarity

Com o "TestSimilarity" é realizado um teste ao módulo "Similarity." Dada a string:

"As armas e os Barões assinalados
Que da Ocidental praia Lusitana
Por mares nunca de antes navegados
Passaram ainda além da Taprobana,
Em perigos e guerras esforçados
Mais do que prometia a força humana,
E entre gente remota edificaram
Novo Reino, que tanto sublimaram"

E a string:

"Novo Reino, que tanto sublimaram
Que da Ocidental praia Lusitana
Por mares nunca de antes navegados
Passaram ainda além da Taprobana,

Em perigos e guerras esforçados
Mais do que prometia a força humana,
E entre gente remota edificaram
As armas e os Barões assinalados"

São criadas as shingles para ambas e é calculada então o índice de similaridade (semelhança) bem como a distância de jaccard entre ambas.

Os valores obtidos são os esperados.

4.7 TesteFalsePositives

A classe "TesteFalsePositive" serve com a razão de o porquê termos selecionado o limite usado na classe principal. Cada uma das chaves das hashmaps é o índice de similaridade e o seu valor pode ser falso, verdadeiro ou filmes encontrados.

4.8 TestBucket

Este teste verifica a congruência do nosso "LSHSimilarity". Cria as possiveis bandas de filmes e imprime-os no terminal para que o utilizador veja as colisões.

Aqui algumas colisões não são fiáveis, enquanto outras o são. Por esse motivo, foi escolhido não utilizar o módulo no trabalho, mas sim decidido fazer referência tanto no código como no relatório para que se compreenda a possível utilização na aplicação.

5 Conclusão

A contribuição acordada por ambos os autores foi de 50% para Mariana Pinto e 50% para Tomás Freitas.

Agradecemos também a contrubuição dos colegas Inês Justo e Luís Moura por nos guiarem na direção certa e pela ajuda prestada em certos módulos, bem como ao professor Carlos Bastos pela extensão da data de entrega.