Relatório do Projeto De Mpei

Raquel Resende Milheiro Pinto,
Rodrigo Lopes Martins

Figura 1: https://www.google.com/search?q=logo+ua+preto

versão 1

Relatório do Projeto De Mpei

Departamento de Eletrónica, Telecomunicações e Informática

Universidade de Aveiro

Raquel Resende Milheiro Pinto,
Rodrigo Lopes Martins
(92948) raq.milh@ua.pt, (93264) rodrigomartins@ua.pt

6/12/2019

Resumo

Neste relatório, descreve-se e apresenta-se a implementação do código desenvolvido para se saber se um determinado jogo tem reviews e se tiver saber quantas é que existe. Também se determina reviews similares e os utilizadores que as fizeram com deteção de spam, mostrando e explicando a maneira como foi implementado.

Para comprovar a integridade do código foram feitos testes, aos varios módulos. Na secção dos resultados permite comprovar o funcionamento do programa.

Conteúdo

Capítulo 1

Introdução

Este trabalho tem como tema o processamento de reviews de jogos, ou seja, saber se determinado jogo tem reviews e se tiver saber quantas, determinar reviews similares e os utilizadores que as fizeram.

Para a determinação de reviews similares usámos algoritmo MinHash e o Local Sensative Hashing (LSH), e para a deteção de spam usamos o algoritmo do Bloom Filter. No inicio para obtermos reviews, tentamos implementar um programa para buscar reviews a vários sites de jogos, mas como não conseguimos usamos um Github com dataSets necessários ao nosso trabalho.

Para formatar os comentários para o padrão desejado foi usado uma biblioteca de java chamada JsonSimple disponível em : https://www.mkyong.com/java/json-simple-example-read-and-write-json/ .

O Bloom Filter foi utilizado para a deteção de comentários idênticos, ou seja, para a deteção de spam.

O algoritmo MinHash é usado para a deteção de similaridade entre documentos, neste caso comentários, logo foi usado para a determinação de reviews similares.

A motivação para este trabalho foi podermos aprofundar o nosso conhecimento sobre a máteria, aplicando assim conhecimentos adquiridos anteriormente pelas aulas desta disciplina.

Neste relatório será explicado a implementação do código que usámos assim como os resultados que obtivemos.

Capítulo 2

Estrutura do trabalho

Este trabalho encontra-se dividido numa classe que interliga os módulos, classes auxiliares, 2 módulos e os seus testes. As classes auxiliares são classes que foram úteis para a configuração dos comentários adquiridos do GitHub e para a criação dos Shingles dos documentos. Os módulos consistem na implementação dos algoritmos de Bloom Filter e de MinHash.

Para verificar o nosso trabalho, execute o ficheiro projeto.java. No terminal aparecerá um menu, no qual pode escolher a funcionalidade do programa que deseja (formatação das reviews, determinação de quantas reviews existe por jogo, deteção de spam - Bloom Filter, deteção de reviews similares - MinHash). Primeiro, antes de começar deverá fazer a formatação das reviews, pois assim obtemos os ficheiros necessários ao funcionamento do resto do programa.

Para testar o Bloom Filter e o MinHash, excute os ficheiros Test_BloomFilter.java e o ficheiro MinHashTest.java, respetivamente.

2.1 Bloom filter

O Bloom Filter consiste basicamente em 3 partes, primeiro iniciamos todas as posições do nosso bloom filter a 0 (no construtor), podemos adicionamos um elemento e também verificar se determinado elemento é membro deste filtro.

Para adicionarmos elementos, calculamos os valores das k funções de dispresão de determinado elemento e atualizamos as posições apropriadas do nosso vetor (chamado ver). Para isso, colocou-se a true as posições devolvidas pelas funções de dispersão.

Na terceira parte da implementação deste modulo, verificamos se um elemento é membro do Bloom Filter, ou seja, se esse elemento já foi adicionado anteriormente. Para isso aplicamos as k funções que usamos para adicionar elementos, mas agora para se verificar se as posições devolvidas pelas funções de dispersão contêm true. Se alguma das posições conter false, significa que o elemento não pertence do conjunto.

Neste projeto, usamos 6 funções de dispresão (k) e o Bloom Filter com

1000000 de tamanho (n). Para este algoritmo, foi implementado testes para ambas as partes, no qual se adicionou um vetor com 7 elementos (m) e a seguir verificamos se eram membros ou não de outro conjunto de elementos.

Calculamos também a taxa de falsos positivos, para Strings geradas aleatoriamente com tamanhos aleatórios. Como se pode ver na imagem seguinte.

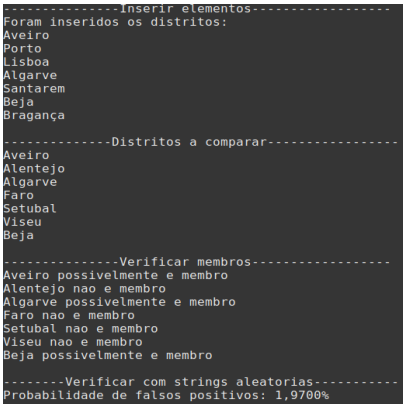

Figura 2.1: Resultados dos testes realizados ao Bloom Filter

2.2 MinHash

Quanto ao MinHash, este foi implementado conforme está descrito nos slides teóricos, ou seja, o MinHash foi implementado em 3 etapas, sendo a primeira a redução dos comentários a Shingles. Foram usados Shingles de dimensão 3. Na segunda etapa criou-se uma matriz booleana que registava com 1 o facto de um determinado comentário possuir um comentário e com 0 caso contrário. Depois foi aplicada a cada linha, caso o valor da posiçao fosse 1, 70 hash functions.

```
//Matriz Booleana
int boolArray[][] = new int[numShingles][count];

for (int j = 0; j < lines.length; j++) {
    Iterator<String> setIterator = set.iterator();

    for (int j2 = 0; j2 < set.size(); j2++) {
        if (lines[j].contains(setIterator.next())) {
            boolArray[j2][j] = 1;
        }else {
            boolArray[j2][j] = 0;
        }
    }
}
```

Figura 2.2: Reviews do jogo Dota 2

Na terceira etapa foi feita a matriz de Assinaturas dos documentos e realizada a comparação entre cada coluna da mesma (cada coluna representa um comentário), depois caso o valor da similaridade das colunas da matriz de assinaturas fosse superior a um valor escolhido pelo utilizador, os comentários seriam considerados similares.

```
    // Criação da Matriz de Assinaturas
    int numHash = 70;
    int hashRows[][] = hashCalc(numHash, numShingles);
    int sigMatrix[][] = new int[numHash][count];

    for (int j = 0; j < numHash; j++) {
        for (int j2 = 0; j2 < count; j2++) {
            sigMatrix[j][j2] = Integer.MAX_VALUE;
        }
    }

    for (int j = 0; j < numShingles; j++) {
        for (int j2 = 0; j2 < count; j2++) {
            if (boolArray[j][j2] == 1) {
                for (int k = 0; k < numHash; k++) {
                    if (hashRows[j][k] < sigMatrix[k][j2]) {
                        sigMatrix[k][j2] = hashRows[j][k];
                    }
                }
            }
        }
    }

    for (int i = 0; i < sigMatrix.length; i++) {
        for (int j = 0; j < sigMatrix[0].length; j++) {
            pwf.print(sigMatrix[i][j] + " ");
        }
        pwf.println();
    }

    return sigMatrix;
}
```

Figura 2.3: Matriz de Assinaturas

Capítulo 3

Resultados

Neste capítulo apresentamos os diversos resultados que obtivemos para sabermos se um determinado jogo tem reviews e quantas é que nele existe, para determinarmos reviews similares e os utilizadores que os fizeram, bem como determinar possivel spam, acompanhados de imagens ilustrativas.

Para sabermos se um determinado jogo tem reviews e quantas, simplesmente implementamos um contador que conta as linhas do ficheiro correspondente ao jogo, pois cada comentário encontra-se numa unica linha desse ficheiro. Na figura 3.1, podemos observar que o resultado para o jogo Dota 2 é de 9720 reviews.

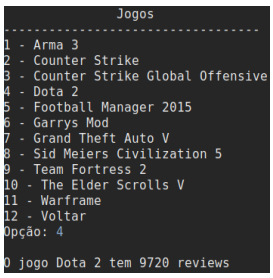

Figura 3.1: Reviews do jogo Dota 2

Para a deteção de similaridade entre comentários usamos o MinHash, mas devido a um pequeno erro na implementação só era possivel identificar os comentários que eram idênticos entre si, depois de resolvido, o algoritmo encontra-se a funcionar sem falhas e deteta comentários semelhantes.

```
82                double similaridade = (double)(similar/sigMatriz.length);
```

Figura 3.2: Código que fazia com que o programa só deteta-se comentários idênticos

Figura 3.3: Erro resolvido

Os comentários semelhantes podem ser visualizados no ficheiro indicado pelo programa no fim de analisar os comentários. O utilizador pode escolher o indice de similaridade com que o programa deve procurar comentários semelhantes.

Quanto a deteção de spam usamos o Bloom Filter, que deteta se determinado comentário é igual a outro, para isso, foi criado um ficheiro com alguns comentários iguais de determinado jogo. Podemos assim dizer que o resultado esperado para a deteção de spam seria o conteúdo do ficherio criado. Como podemos observar nas imagens para o exemplo do jogo Arma 3.

```
MightyMuffin;-;best game i have ever played. BEST AND WORST GAME THO. your pc will take a hit from this gamein the fps range
but over all my fav game
kony;-;ArmA2 has much more content and is far more realistic (only medic can heal and it takes a while, weapons acting as real
life counterparts, in ArmA3 everything is 'balanced' and you can self-heal in 3 seconds by clicking a proper button). ArmA2
has also more vehicles, more weapons, more everything... really, there is nothing to go for ArmA3 except a bit better graphic,
but this is hardly anything important, especially because ArmA2 looks not much worse. Moreover it's been over three months
since the release date and the game still has unfinished campaign and terrible AI bugs. You don't have to take my word for it,
watch a video: http://steamcommunity.com/sharedfiles/filedetails/?id=178199064
KB_717;-;Arma 3 is a war simulator but gets used for a lot of other reasons example, wasteland altis life etc but there are a
few problems with it as it still just been released example the FPS on servers are very bad and i know its not my computer as
many other people has complained about it but overall it is worth the $60
dupeblow;-;Good PC? Buy Arma 3. its worth it
Kyuu;-;UNMODDED: Oh god it hurts. Ignoring  LP bait mods like Altis Life, Wasteland and DayZ. Arma 3 modded is one of the most
enjoyable milsims out there and I fully recommend it as long as you don't mind spending first few hours hooking up mods and
scenarios which make the game infinitely more fun with huge amount of replay value.
jABRiL;-;mod breaking point dont miss it
hellscoop;-;THE ROCKS!!!! THE ING ROCKS!!!!! I SWEAR THEY ARE OUT TO GET ME!!!!!!!!!!
Nitromen23;-;Arma 3 is very glitched ad horribly unoptimized and yet is still an amazing game so i will rate it a 4 and
reccomend it!
IntoTheGabe;-;So once you've figured out what the  you're doing you'll like this game, if not then go play some Call of Duty.
IB6UB9;-;Reminds me a lot of DayZ. It doesn't have any zombies, and it isn't as pick up and play as Day Z either. 10/10-IGN
ola;-;kjuihuyftyddtrfhdsrgstuhiojojnjnjwejk.hwerbukeitugyq7ilwuknjnioqpwkxlwmklvnepruipehw
```

Figura 3.4: Ficheiro criado com alguns comentários copiados

```
Comentario criado por MightyMuffin:
best game i have ever played. BEST AND WORST GAME THO. your pc will take a hit from this game in the fps range but
over all my fav game -> possivelmente spam
Comentario criado por kony:
ArmA2 has much more content and is far more realistic (only medic can heal and it takes a while, weapons acting as
real life counterparts, in ArmA3 everything is 'balanced' and you can self-heal in 3 seconds by clicking a proper
button). ArmA2 has also more vehicles, more weapons, more everything... really, there is nothing to go for ArmA3
except a bit better graphic, but this is hardly anything important, especially because ArmA2 looks not much worse.
Moreover it's been over three months since the release date and the game still
has unfinished campaign and terrible AI bugs. You don't have to take my word for it, watch a video:
http://steamcommunity.com/sharedfiles/filedetails/?id=178199064 -> possivelmente spam
Comentario criado por KB_717:
Arma 3 is a war simulator but gets used for a lot of other reasons example, wasteland altis life etc but there are
a few problems with it as it still just been released example the FPS on servers are very bad and i know its not
my computer as many other people has complained about it but overall it is worth the $60 -> possivelmente spam
Comentario criado por dupeblow:
Good PC? Buy Arma 3. its worth it -> possivelmente spam
Comentario criado por Kyuu:
UNMODDED: Oh god it hurts. Ignoring  LP bait mods like Altis Life, Wasteland and DayZ. Arma 3 modded is one of
the most enjoyable milsims out there and I fully recommend it as long as you don't mind spending first few
hours hooking up mods and scenarios which make the game infinitely more fun with huge amount of replay value.
-> possivelmente spam
Comentario criado por jABRiL:
mod breaking point dont miss it -> possivelmente spam
Comentario criado por hellscoop:
THE ROCKS!!!! THE ING ROCKS!!!!! I SWEAR THEY ARE OUT TO GET ME!!!!!!!!!! -> possivelmente spam
Comentario criado por Nitromen23:
Arma 3 is very glitched ad horribly unoptimized and yet is still an amazing game so i will rate it a 4 and
reccomend it! -> possivelmente spam
Comentario criado por IntoTheGabe:
So once you've figured out what the  you're doing you'll like this game, if not then go play some Call of Duty.
-> possivelmente spam
Comentario criado por IB6UB9:
Reminds me a lot of DayZ. It doesn't have any zombies, and it isn't as pick up and play as Day Z either. 10/10-IGN
-> possivelmente spam
```

Figura 3.5: Ficheiro de comentários considerados spam

Para a deteção de spam também poderia ser utilizado o MinHash, pois se o utilizador decidir que o indice de similaridade seja 1, o programa vai encontrar todos os comentários identicos entre si, ou seja, vai encontrar spam.

Capítulo 4

Conclusão

Este trabalho, apesar de ter sido trabalhoso, aprendemos a trabalhar melhor com estes algoritmos (Bloom Filter e MinHash), como base o que aprendemos nas aulas práticas e teóricas.

Ao longo deste projeto, tivemos vários problemas e com ajuda dos nossos docentes e de alguns colegas mais velhos, conseguimos ultrapassar isso. Concluindo, trabalhamos bem em conjunto e conseguimos apresentar o nosso projeto as funcionalidades pretendidas. Foi fornecido uma classe intitulada "Teste"com o algoritmo de MinHash com LSH que não foi usado pois não produzia resultados, devido ao erro, que também estava presente no MinHash, que foi apenas descoberto no fim do projeto estar concluido. Neste trabalho os integrantes tiveram uma contribuição de 50% cada um.

4.1 Bibliografia

https://www.sanfoundry.com/java-program-implement-bloom-filter/ -> (Usado para ajudar no Bloom Filter)

https://github.com/mulhod/steam_reviews/tree/master/data -> (GitHub dos comentários)

https://stackoverflow.com/questions/47407251/optimal-way-to-find-next-prime-number-java -> (Função de gerar numeros primos)

https://respostas.guj.com.br/44574-gerar-string-aleatoria-com-ramdom -> (Gerar strings aleatorias)

https://www.devmedia.com.br/forum/funcao-que-gera-string-aleatoria/568082 -> (Gerar strings aleatorias)