Métodos Probabílisticos para Engenharia Informática

Relatório: Sistema de Análise, Classificação e Recomendação de Jogos

Eduardo Alves: nºmec 104179
Joaquim Martins: nºmec 115931

eduardoalves@ua.pt
joaquimmartins33@ua.pt

Universidade de Aveiro
2024

1 Introdução

O objetivo deste projeto é desenvolver, testar e demonstrar uma aplicação prática que combine técnicas de classificação, filtragem e detecção de itens similares. Para isto, a aplicação deve integrar três componentes principais: um classificador Naïve Bayes, um Bloom Filter e um sistema para determinação de itens similares baseado em Minhash. O projeto será desenvolvido em etapas bem definidas, garantindo que cada módulo seja criado de forma eficiente, testado cuidadosamente e, por fim, integrado numa demonstração prática. Todo este trabalho será realizado em MATLAB.

Para o nosso projeto, vamos fazer um sistema de análise, classificação e recomendação de jogos interativo. Onde o utilizador ao executar o ficheiro main.m (Ficheiro responsável pela aplicação conjunta) poderá selecionar entre várias opções no Menu Principal.

- **Naive Bayes**: O utilizador seleciona tags e atribui peso de importancia das mesmas, o sistema retorna se é mais provável o conjunto de tags pertencer à classe de Menor ou maior de 18 anos.
- **Bloom Filter**: O utilizador deve selecionar um jogo e uma tag das múltiplas opções disponiveis, o Bloom Filter deve retornar se a tag pertence ou não ao jogo.
- **MinHash**: O utilizador escolhe um jogo normal e o sistema recomenda de volta um jogo VR parecido. Para fazer isto, este usa a semelhança entre as tags disponíveis.
- **Testes dos Módulos**: O utilizador escolhe entre os módulos para testar. O sistema executa o ficheiro de testes escolhido pelo utilizador.
- **Conjunto de Listas**: Acesso direto aos Datasets.
- **Sair**: Encerra o Programa

2 Como está dividido o código?

Programa	**Funcionalidade**
main.m	Demontração conjunta dos vários algorítmos
naivebayes.m	Código relativo ao módulo desenvolvido "Naive Bayes"
bloomfilter.m	Código relativo ao módulo desenvolvido "Bloom Filter"
minhash.m	Código relativo ao módulo desenvolvido "MinHash"
naivebayes_testes.m	Testes do módulo "Naive Bayes"
bloomfilter_testes.m	Testes do módulo "Bloom Filter"
minhash_testes.m	Testes do módulo "MinHash"
Funções auxiliares do Bloom Filter	Bloom_Filter.m; Bloom_Filter_insert.m; Bloom_Verify.m; string2hash.m
Funções auxiliares da MinHash	generateMinHash.m
jogos_normais.xlsx	Dataset dos jogos de consola e PC
jogos_VR.xlsx	Dataset dos jogos VR
naivebayes_data_treino.xlsx	Dataset para Naive Bayes

Table 1: Divisão de código

3 Naive Bayes

Naive Bayes é um algoritmo de aprendizagem supervisionada fundamentado no teorema de Bayes, amplamente utilizado para problemas de classificação. Esse modelo parte do princípio de que as variáveis preditoras, ou características, são independentes umas das outras. Este algoritmo tem se mostrado bastante eficiente em várias tarefas de classificação, especialmente quando lidamos com grandes volumes de dados ou com variáveis binárias, como é o caso do nosso projeto, que visa classificar jogos.

3.1 Aplicação no nosso Projeto

No nosso projeto, o Naive Bayes tem como objetivo classificar jogos como adequados para maiores ou menores de 18 anos. Para este fim, utilizamos uma tabela Excel com dados organizados por tags ('features'), onde cada linha representa um conjunto de características que descrevem os jogos. As tags incluem categorias como "Survival", "RPG", "Horror", "Multiplayer", entre outras, que indicam características específicas dos jogos. Cada coluna da tabela corresponde a uma dessas tags, e os valores são binários (0 ou 1), onde 1 significa que a tag está presente e 0 indica a sua ausência. A última coluna da tabela representa a classificação etária dos jogos, com duas classes possíveis: maior de 18 anos ou menor de 18 anos. Essa classificação é determinada com base nas combinações de tags, onde características como "Horror", "Tiro", "Dark Atmosphere", entre outras, podem indicar que o jogo é adequado para maiores de 18 anos. Já jogos com tags como "Party Games" ou "Criatividade" tendem a ser classificados como "menores de 18 anos".

3.2 Como é que o utilizador interage com o sistema?

Ao selecionar a opção no menu, é pedido ao utilizador que escolha as tags que mais lhe interessam, com base nas suas preferências pessoais. Após selecionar as tags desejadas, o utilizador deve atribuir um peso de importância a cada uma delas, indicando o quão relevante cada tag é para si. Com essas informações, o modelo de Naive Bayes utiliza os pesos atribuídos e as combinações de tags para determinar se o jogo é mais indicado para maiores de 18 anos ou menores de 18 anos. A classificação final é baseada não apenas na presença das tags, mas também nos pesos atribuídos pelo utilizador, o que permite uma recomendação mais personalizada.

3.3 Testes do Módulo Naive Bayes

Para garantir o desempenho do modelo, dividimos os dados disponíveis em dois conjuntos: 70% dos dados foram reservados para o treino do modelo e os restantes 30% foram utilizados para testagem. Essa divisão não só garantiu que o modelo seja avaliado de forma justa mas também que seu desempenho seja medido com dados que o modelo não tenha usado durante o seu processo de treino. Para correr os ficheiros de teste do módulo Naive Bayes, basta executar o programa "naivebayes testes.m" ou executar o main, selecionar a opção "Testar os Módulos" e de seguida "Testar Naive Bayes".

4 Bloom Filter

O Bloom Filter é uma estrutura de dados probabilística projetada para determinar se um elemento pertence a um conjunto, Este modelo funciona por meio de funções hash que mapeiam elementos para posições específicas nm vetor de bits, assumindo o risco de falsos positivos em troca de economias significativas de espaço.

4.1 Aplicação no nosso Projeto

No contexto do nosso projeto, o Bloom Filter é utilizado para verificar de forma eficiente se uma certa tag pertence a um determinado jogo. Para isto, o programa acede ao Dataset de jogos para consolas e PCs (jogos_normais.xlsx). Com os dados disponiveis neste dataset, o programa identifica e separa os diferentes nomes dos jogos e tags. De seguida, o programa verifica a existência de um bloom filter para cada jogo. Caso ainda não exista, este é inicializado, e apenas as tags com o valor de 1 nas tabelas são inseridas no Bloom filter, usando a função auxiliar Bloom_Filter_insert. Quando um jogo e tag são selecionados, o Bloom Filter desse jogo é recuperado, e é feita a verificação da tag no filtro através da função auxiliar Bloom_Verify.

4.2 Como é que o utilizador interage com o sistema?

Ao selecionar a opção no Menu, é pedido ao utilizador para selecionar um jogo e de seguida uma tag, entre os dados disponiveis na tabela jogos_normais.xlsx. O Bloom filter começa então o processo de verificação se a tag pertence ou não ao jogo. Caso pertence, o sistema retorna uma mensagem a indicar que a tag provávelmente pertence ao jogo. Caso não pertença, o Bloom Filter garante que a tag não pertence ao jogo e o sistema informa o utilizador.

4.3 Testes do Módulo Bloom Filter

Ao executar o ficheiro de testes bloomfilter_teste, o programa averigua se cada jogo já tem um filtro de Bloom associado, caso não tenha, este é criado e atribuido. Após isto, as tags que pertencem a cada jogo (ou seja, que correspondem a 1 na tabela xlsx) são associadas aos filtros de Bloom correspondentes. No inicio do ficheiro é também criado um mapa utilizado para armazenar as tags, que realmente correspondem ao jogo, associadas a cada jogo. Após inserir as tags, o código compara se as tags presentes no mapa correspondem às presentes no filtro de Bloom. Caso uma tag seja detetada como existente no filtro de bloom, mas não esteja presente no mapa é considerado um Falso Positivo. Ao detetar estes falsos positivos, o sistema faz print do Jogo e da Respetiva tag que não deviam pertencer ao conjunto. O programa mantém também uma contagem do número total de verificações, número total de tags inseridas no filtro de bloom. número de falsos positivos, percentagem de falsos positivos e a percentagem da probabilidade de ocorrer falsos positivos, utilizando a fórmula presente nos slides da disciplina:

$$[1 - (1 - 1/n)^{(}km)]^k$$

Para correr o programa de bloomfilter_teste, basta executar (dar Run) diretemante no ficheiro ou executar o ficheiro main e selecionar a opção "Testar os Módulos" e de seguida "Testar Bloom Filter".

5 MinHash

O MinHash é uma técnica probabilística amplamente utilizada para estimar a similaridade entre dois conjuntos de dados. É um método que se baseia em criar assinaturas compactas de conjuntos utilizando funções hash, o que permite comparar volumes elevados de dados de maneira eficiente. Na MinHash, a similaridade entre dois conjuntos de dados é medida através da similaridade de Jaccard, que calcula a proporção que são comuns entre os dois conjuntos. Esta técnica probabilistica usa a comparação de assinaturas reduzidas, evitando comparações de conjuntos completos de dados. Isto não só diminiu o tempo de processamento dos dados, como economiza espaço.

5.1 Aplicação no nosso Projeto

O minHash é usada no nosso projeto para, através das tags dos jogos normais, recomendar o jogo VR com as tags mais semelhantes. Ou seja, explicado de uma maneira mais abrangente, recomenda os jogo VR mais semelhantes com o jogo normal escolhido pelo utilizador. Este processo começa por gerar e atribuir assinaturas minHash para todos os jogos normais e VR, cujo objetivo é calcular a proporção de hashes iguais entre as assinaturas dos dois jogos. Após isto, os jogos são ordenados por ordem consoante o seu nível de similaridade com o jogo escolhido pelo utilizador (do mais semelhante ao menos semelhante).

5.2 Como é que o utilizador interage com o sistema?

O utilizador começa por escolher a opção de sugestão de um jogo VR baseado num jogo normal no Menu. De seguida é pedido ao utilizador para selecionar um dos vários jogos normais disponiveis. O sistema utiliza as assinaturas geradas pelas funções hash para calcular a semelhança de Jaccard. De seguida retorna ao utilizador o jogo VR mais semelhante. Após cada recomendação, o sistema pergunta ao utilizador se ele deseja receber uma nova recomendação. Caso o utilizador opte por continuar, o próximo jogo mais semelhante será recomendado com base na lista ordenada de similaridades. Caso ele não opte por continuar, o programa informa que as recomendações foram encerradas pelo utilizador e o programa termina. Ao fim de 5 recomendações seguidas, o limite de recomendações é atingido e o programa termina.

5.3 Testes do Módulo MinHash

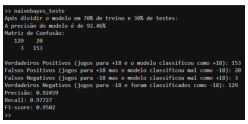
No ficheiro de testes minhash_teste.m, o programa pede ao utilizador para escolher um jogo normal. Com esta informação, o sistema faz o uso de assinaturas hash geradas para calcular a similaridade de jaccard aproximada (minhash). De seguida o programa calcula a similaridade de jaccard real usando a fórmula:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

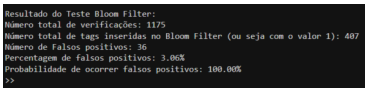
e faz a comparação das mesmas. O programa de teste imprime também de forma ordenada do mais para o menos similar os jogos VR recomendados para o jogo escolhido com a respetiva similaridade Minhash. Para correr o programa, basta dar run no ficheiro ou executar o ficheiro main e selecionar "Testar os Módulos" e "Testar Minhash".

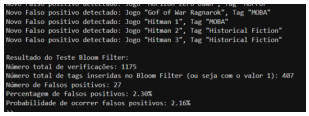
6 Análise dos Resultados obtidos durante os Testes

Os testes do Naive Bayes foram positivos. Após dividirmos os dados em 70% para treino e 30% para testes, calculámos a precisão do modelo de 92,46%, comparando os valores de testes com os valores previstos. Obtivemos também a matriz de confusão para os dados de teste do nosso programa, ao qual obtivemos 129 Verdadeiros positivos, 20 Falsos positivos, 3 Falsos Negativos e 153 Verdadeiros Negativos. Após isto, calculámos com estes valores, as métricas de avaliação: Accuracy, Recall e F1-Score, nas quais obtivemos 0.92459, 0.97727 e 0.9502, respectivamente. As vantagens da nossa implementação são a elevada eficiência e desempenho, no entanto supõe que as variáveis de entrada são independentes.



Os testes do filtro de Bloom foram positivos. Usámos a fórmula presente nos slides teóricos para calcular a probabilidade de falsos positivos e transformámos-la em percentagem. Este ficheiro de testes alertou-nos também para corrigirmos os valores de k (Número de funções hash) e N (Tamanho do Bloom Filter), visto que previamente tinhamos uma taxa muito elevada de falsos positivos. Após calcular o k óptimo e ajustar o tamanho do filtro para um tamanho de 8x os dados inseridos no filtro, obtivemos uma chance de 2.16% de ocorrência de Falsos Positivos. Inserimos dados de testes, que não tinham sido previamente usados, adicionando de forma aleatória uma tag nova, e o programa detetou entre 2% e 3% de falsos positivos, identificando os jogos e as respetivas tags falso positivos. Contabilizamos também o total de tags verificadas, inseridas no filtro e número de falsos positivos, permitindo calcular a percentagem de falsos positivos ocorridos. As vantagens da implementação do nosso filtro de bloom é que este é altamente eficiente em termos de uso de memória mas corre o risco de Falsos Positivos. A imagem abaixo à esquerda mostra os valores antigos, e a da direita, os novos.





Os resultados da testagem da MinHash também foram positivos, comparámos a similaridade de Jaccard aproximada (MinHash) com a similaridade de Jaccard real, calculamos a distância entre ambas e a respetiva percentagem de distância para o conjunto de todos os jogos VR baseados na escolha de um jogo normal, ao qual conseguimos analisar valores muito próximos, o que demostra uma boa configuração da minhash implementada. De seguida testámos também, a ordenação por ordem de similaridade de Jaccard aproximada (Minhash), ao qual conseguimos analisar que foi feita de forma ordenada e correta, consoante os valores das tags e os valores pretendidos. A implementação é eficiente e aproxima-se da similaridade de Jaccard real, mas gera apenas estimativas, com limitações em conjuntos pequenos como o nosso. A imagem abaixo à esquerda a comparação de similaridades, e a da direita, as recomendações ordenadas.