universidade de aveiro

Trabalho Prático-2024/2025
Métodos Probabilísticos para Engenharia Informática

Sistema de recomendação de música

Turma P4:
Prof. Carlos Bastos

João Su - 112790
Gabriel Boia - 113167

Introdução

Este projeto desenvolve um sistema de recomendação de músicas baseado em 3 técnicas abordadas nas aulas práticas e teórico-práticas de Métodos Probabilísticos para Engenharia Informática.

As 3 técnicas utilizadas neste projeto são:

Classificador Naive Bayes, utilizado para classificar documentos consoante probabilidades condicionais, provenientes da ocorrência de palavras, o classificador vai ser utilizado na lista de músicas recomendadas de forma a determinar se o utilizador gosta (LIKE) ou não delas (DISLIKE).

Filtro Bloom, utilizado para filtrar a base de dados de músicas a ser sugeridas de forma, não sejam recomendadas músicas já na playlist do utilizador e também para filtrar músicas repetidas dentro dos próprios datasets utilizados.

MinHash, a técnica de MinHash é aplicada para medir a similaridade entre conjuntos. Com base nessa abordagem, a partir de datasets de playlists de diferentes utilizadores, identifica-se a playlist mais semelhante à playlist do utilizador, permitindo recomendar músicas relevantes que ainda não estão na coleção do mesmo.

Nas próximas páginas vamos apresentar uma descrição de como foram implementadas estas técnicas em MATLAB e como foram efetuados os devidos testes para verificar as suas funcionalidades, vamos também partilhar os resultados obtidos desses testes e descrever sucintamente como foi criada uma demonstração da utilização conjunta destas 3 técnicas que possibilitam o funcionamento do sistema de recomendação de músicas, realçando os seus limites e as suas vantagens.

Naive Bayes

O módulo que implementa o classificador Naive Bayes encontra-se no ficheiro NaiveBayes.m, o módulo está implementado como uma classe onde se encontram definidas todas as funções que implementam a funcionalidade do classificador.

Este módulo é testado no ficheiro NaiveBayesTest.mlx onde o classificador é treinado e testado com um dataset previamente fabricado em python. Neste ficheiro podemos observar resultados de métricas importantes para avaliar a performance do classificador como precisão, recall, f1 e accuracy. Todas as funções da classe foram testadas neste ficheiro, para verificar os testes basta executar o ficheiro NaiveBayesTest.mlx.

Bloom Filter

O módulo que implementa o Filtro de Bloom encontra-se no ficheiro BloomFilter.m, o módulo está implementado como uma classe onde se encontram definidas todas as funções que implementam a funcionalidade do filtro.

Este módulo é testado no ficheiro filtro_bloom.mlx onde o filtro é testado com um dataset já existente. No filtro foram inseridos os valores do ficheiro “playlist.csv”, playlist do utilizador, e o resultado é o dataset anteriormente referido filtrado. Neste teste podemos observar os resultados no ficheiro criado no final do teste “dataset_filtrado_final.csv”. Todas as funções da classe foram testadas neste ficheiro, para verificar os testes basta executar o ficheiro filtro_bloom.mlx.

MinHash

O módulo que implementa a deteção de conjuntos similares usando MinHash encontra-se no ficheiro MinHash.m, o módulo está implementado como uma classe onde se encontram definidas todas as funções necessárias para a deteção de itens semelhantes.

Este módulo é testado no ficheiro min_hash.mlx onde o filtro é testado com vários conjuntos diferentes obtidos a partir de python. Para o teste foram utilizadas 5 playlists diferentes (incluindo a playlist do utilizador) e o resultado é uma matriz da distância entre cada uma das playlists. Neste teste podemos observar os resultados na matriz distSign criada no final do mesmo. Todas as funções da classe foram testadas neste ficheiro, para verificar os testes basta executar o ficheiro min_hash..mlx.

Demonstração

A demo do projeto centra-se numa combinação dos 3 módulos descritos anteriormente para implementar um sistema de recomendação de músicas.

Para este exemplo consideramos uma persona que tem gosto apenas por música do género metalcore.

Primeiro começamos por criar um filtro de bloom onde são introduzidas as músicas do utilizador, usamos o filtro para remover músicas que já estão na playlist deste utilizador do dataset de músicas.

Em seguida, tendo em conta as 4 playlists que foram criadas para simular outros utilizadores utilizamos a técnica para deteção de itens similares com MinHash de forma a calcular a playlist mais semelhante à do utilizador.

Por fim, escolhemos aleatoriamente músicas da playlist mais semelhante à do utilizador e também do dataset de músicas filtrado inicialmente formando uma lista de músicas recomendadas, passamos esta lista uma última vez pelo filtro de bloom para remover músicas repetidas e obtemos assim a lista final de músicas recomendadas para o utilizador.

Limitações

Apesar de o programa ser funcional na recomendação de músicas aos utilizadores, existem limitações inerentes às técnicas utilizadas que afetam a precisão e a confiabilidade dos resultados:

Apesar do programa conseguir recomendar músicas aos utilizadores, não conseguimos garantir que as recomendações sejam sempre as mais corretas já que o classificador apresenta uma accuracy de cerca de 56%.

O Filtro de Bloom, devido à sua natureza probabilística, apresenta a seguinte limitação: Quando indica que um valor está presente, existe uma probabilidade de falso positivo, o que significa que não podemos garantir que a música removida já existia realmente na coleção do utilizador.

Já a deteção de semelhanças é feita entre todos os conjuntos, mesmo que uma playlist apresente a menor distância em relação à playlist do utilizador, não podemos assegurar que as músicas sugeridas dessa playlist sejam realmente as mais semelhantes às preferências do utilizador.

Vantagens

Com o Classificador Naive Bayes conseguimos estimar se o utilizador irá ou não gostar da música recomendada para melhores recomendações.

O Filtro de Bloom permite verificar rapidamente e com certeza se a música não está na playlist do utilizador e estimar se a música já se encontra na playlist do utilizador.

O algoritmo de deteção de semelhanças entre conjuntos com MinHash permite reconhecer playlists semelhantes ao do utilizador para recomendar músicas com uma probabilidade alta de serem do gosto do utilizador.