universidade de aveiro deti departamento de electrónica, telecomunicações e informática

Métodos Probabilísticos para Engenharia Informática

Guião PL04

Professor:

Carlos Bastos(*cbastos@ua.pt*)

Amaro Sousa(*asousa@ua.pt*)

Realizado por:

Daniel Madureira, 107603, P4

José Gameiro, 108840, P4

Índice

1. Introdução

No âmbito da cadeira de Métodos Probabilísticos para Engenharia Informática, foi-nos proposto desenvolver uma aplicação em Matlab com algumas funcionalidades de um sistema online de disponibilização de filmes. A aplicação considera um conjunto de utilizadores identificados por um ID e um conjunto de filmes também identificados por um ID, ambos os ID's têm de ser números inteiros positivos.

Foi nos disponibilizado um ficheiro u.data que contém vários dados, que se encontram organizados por várias colunas, os filmes visualizados por um determinado utilizador. Para esta apliacação serão necessários os dados das duas primeiras colunas que são os ID's dos utilizadores e os ID's dos filmes que cada utilizador visualizou.

Também nos foi disponibilizado um ficheiro films.txt que, que contém as seguintes informações sobre filmes:

- O nome e o ano do filme;

- Os géneros do filme, sendo que cada filme pode ter no máximo 6 géneros.

Durante este relatório iremos explicar os métodos e funções que utilizámos para construir a nossa aplicação.

2. Desenvolvimento

Para a implementação desta aplicação criámos dois ficheiros Matlab:

- readData.m: este ficheiro tem como objetivo carregar os dados dos ficheiros u.data e guardá-los num cell array, em que cada cell represente um utilizador e os dados que se encontram nesse cell são os filmes visualizados pelo utilizador e films.txt, que guardámos também num cell array, sendo que cada, na primeira coluna, em cada cell, estão contidos os nomes dos filmes e nas colunas dois a sete estão presentes os diferentes estilos de cada filme. Neste ficheiro também é necessário guardar em variáveis as matrizes de assinaturas resultantes da aplicação de diferentes métodos de MinHash, consoante as opções;

- main.m: este ficheiro terá o objetivo de comunicar com o utilizador, ou seja, disponibilizar as opções que a aplicação contém e o código para cada opção.

Para o ficheiro readData.m tivemos que desenvolver 3 métodos de MinHash, que iremos explicar agora cada um deles.

2.1. minhash.m

Esta função minHash tem como objetivo retornar uma matriz de assinaturas com os vetores MinHash que correspondem ao conjunto de filmes avaliados por cada utilizador.

Relativamente à implementação, em termos de código Matlab, esta função tem como entrada um conjunto de dados (que neste caso é uma lista de filmes assistidos pelos utilizadores) e o número de funções hash a utilizar. A função retorna uma matriz de assinaturas, onde cada linha representa um utilizador e cada coluna representa uma hash function.

O código inicializa uma matriz de assinaturas com números infinitos em cada posição e depois cria uma barra de progresso. Em seguida, para cada utilizador, é calculado o número de filmes que ele assistiu e, para cada filme, é gerado um vetor de hash codes através da função DJB31MA_mod. Depois, o vetor de hash codes é comparado com o vetor de assinaturas do utilizador atual e o menor valor é escolhido para cada coluna. Depois de percorrer todos os utilizadores e filmes, a barra de progresso é fechada.

2.2. minhash_moviesStyles.m

Esta função deve de retornar uma matriz de assinaturas com os vetores MinHash correspondentes ao conjunto de géneros cinematográficos de cada filme.

Em relação ao código Matlab, esta função começa por criar uma matriz de assinaturas para cada filme, usando o algoritmo de MinHash. A matriz de assinaturas terá o mesmo número de linhas que o número de filmes e o número de colunas será igual ao número de funções hash especificado.

Para cada filme, a função percorre todas as suas posições de estilo (desde a posição 2 até a 7) e, se o estilo existir (não for vazio), a função cria um conjunto de códigos hash para esse estilo usando a função DJB31MA_mod. Em seguida, atualiza a linha correspondente na matriz de assinaturas com o menor valor entre os códigos hash criados e os valores já presentes na matriz.

A função também exibe uma barra de progresso para mostrar o progresso do processamento. Quando o processamento estiver concluído, a função retorna a matriz de assinaturas criada.

Terminamos assim a explicação das funções de MinHash que foram utilizados no ficheiro readData.m, a partir de agora iremos explicar o código presente no ficheiro main.m e noutras funções utilizadas neste ficheiro.

2.3. getSimilares.m

Esta função tem como objetivo encontrar os pares de utilizadores similares através do cálculo da distância de Jaccard entre as suas assinaturas (que são vetores gerados através do algoritmo MinHash). A distância de Jaccard é calculada como a divisão entre as duas assinaturas dividida pelo número de hash functions utilizadas. Se a distância de Jaccard for menor que o threshold (limiar) especificado, os dois utilizadores são considerados similares e são adicionados ao array SimilarUsers. O código também inclui uma barra de progresso e uma medida do tempo de execução.

2.4. selectMovies.m

Esta função cria uma matriz de assinaturas para cada estilo de filme, que tem o número de linhas igual ao número de filmes e o número de colunas igual ao número de hash functions especificado. Cada linha representa um filme e cada coluna representa uma função de hash.

Para cada filme, a função verifica se existe algum estilo definido nas posições 2 a 7 da célula de dados do filme. Se existir, a função aplica a função de hash DJB31MA_mod ao estilo e atualiza a linha da matriz de assinaturas com os valores mínimos das funções de hash. Isso é feito para cada estilo do filme, e no final cada linha da matriz de assinaturas representa o conjunto de estilos do filme.

Uma barra de progresso é exibida enquanto a função é executada. Quando a função termina, a barra de progresso é fechada.

2.5. main.m

Tal como referido anteriormente esta função vai buscar as variáveis guardadas no ficheiro readData e começa por pedir um ID de um utilizador. Este ID deve de ser um número entre 1 e 943 e caso o ID introduzido não seja válido é pedido novamente que se introduza um ID. Caso o ID seja válido avança para a próxima instrução.

É impresso no terminal as opções disponíveis e pede-se ao utilizador que insira uma:

Caso seja a opção 1 é apresentado ao utilizador o título dos filmes que ele já assistiu e, para isso, o código percorre os filmes visualizados pelo utilizador e vai buscar o título do filme no cell array data, que contém os títulos e estilos dos filmes. Depois são impressos os títulos dos filmes já visualizados pelo utilizador escolhido, juntamente com o seu ID.

Se a opção escolhida for a 2, esta terá como objetivo sugerir filmes ao utilizador baseando-se nas avaliações de outros utilizadores que sejam similares a ele.

Primeiro, é calculado o número total de utilizadores e é definido um threshold de similaridade (neste caso, 0.4). Também é definido o número de hash functions a serem utilizadas (neste caso, 5). Em seguida, é chamada a função getSimilaresHash que retorna uma matriz com os pares de utilizadores mais similares e as suas distâncias de Jaccard.

Depois, é selecionado os dois utilizadores mais similares. Em seguida, é criado um array vazio chamado allMovies para armazenar os filmes avaliados pelos dois utilizadores mais similares, para isso, é percorrido o array most2_similarusers e, para cada utilizador, são obtidos os filmes que ele avaliou (através dos dados presentes na variável movies). Estes filmes são adicionados ao array allMovies.

O array allMovies é depois filtrado para eliminar filmes duplicados (através da função unique). Em seguida, é obtida a lista de filmes avaliados pelo utilizador escolhido (através da função movies) e, por fim, é chamada a função setdiff para encontrar os filmes que o utilizador escolhido não avaliou, mas que os dois utilizadores mais similares avaliaram.

Por fim, é impresso o nome de cada um destes filmes não avaliados pelo utilizador escolhido, mas avaliados pelos dois utilizadores mais similares.

Caso o utilizador selecione a opção 3 é sugerido um conjunto de filmes baseados nos filmes já avaliados pelo utilizador escolhido.

Na implementação do código, o primeiro passo consiste em escolher os filmes já avaliados pelo utilizador inserido, através da variável "rated_movies". Depois, é chamada a função "selectMovies", que irá retornar um array de filmes que têm uma distância de Jaccard menor que 0.8 em relação aos filmes avaliados pelo utilizador e que ainda não foram avaliados pelo mesmo. Esse array é guardado na variável "selected_movies".

Depois, é removido qualquer filme repetido do array "selected_movies" através da função "unique", e é criado um array "movie_count" para guardar o número de vezes que cada filme aparece no array "selected_movies".

Finalmente, é encontrado o filme que aparece mais vezes no array "selected_movies", através da função "max", e também é encontrado o segundo filme que aparece mais vezes, através da função "max" e do operador "~=". Depois, os nomes desses dois filmes são impressos na tela.

Na opção 4 o código que nós implementámos encontra-se incompleto e tem como objetivo procurar filmes em que o nome contenha uma determinada string. O objetivo é encontrar filmes similares àquele cujo nome foi inserido pelo utilizador. Para isso, são criados shingles (sequências de caracteres) a partir do nome dos filmes e dos caracteres inseridos pelo utilizador. Depois, são aplicadas as funções hash DJB31MA aos shingles, para encontrar os valores hash correspondentes. Em seguida, é encontrada a distância de Jaccard entre os valores hash dos filmes e do input do utilizador, e os filmes cuja distância de Jaccard seja menor que uns determinados limiares são selecionados. Finalmente, os filmes selecionados são classificados por ordem de distância de Jaccard e os cinco primeiros são mostrados ao utilizador.

3. Conclusão

Em conclusão, com o desenvolvimento desta aplicação em Matlab, expandimos os nossos conhecimentos relativamente à programação em Matlab e aos algoritmos MinHash que foram necessários usar. Desenvolvemos um script Matlab com 4 opções funcionais usando métodos MinHash. Devido à falta de tempo não conseguimos acabar a implementação da opção 4, logo esta não funciona totalmente.