Documentação *Programming Assignment* 1

Gabriel Soares da Silva
Universidade Federal de Minas Gerais
Belo Horizonte, Minas Gerais

1 INTRODUÇÃO

O objetivo deste trabalho é implementar um recomendador de filmes utilizando técnicas de filtragem colaborativa. De modo geral, filtros colaborativos fazem a predição dos interesses de um usuário alvo com base nas informações a respeito de outros usuários. A ideia base é a de que se uma pessoa *A* tem a mesma opinião de uma pessoa *B* a respeito de um item, é mais confiável que as opiniões de *A* em outros assuntos sejam semelhantes as de *B* do que as opiniões de uma outra pessoa aleatória.

Os filtros colaborativos podem ser classificado em três tipos: filtros baseados em memória, onde o filtro utiliza os dados de avalização para computar a similaridade entre itens ou entre usuários; filtros baseados em modelo, onde o filtro utiliza técnicas de mineração de dados, algoritmos de aprendizado de máquina, entre outras coisas para tentar prever a nota do usuário; e modelos híbridos, que combinam filtros baseados em memória e filtros baseados em modelo para calcular as predições.

No trabalho atual, o programa deverá ser executado por meio do comando

```
./recommender ratings.csv targets.csv > submission.csv
```

onde o arquivo *rating.csv* é o histórico do sistema e contém tuplas do tipo{*user, item, rating*}- onde os valores representam, respectivamente, o identificador do usuário, o identificador do item e a nota dada pelo usuário ao item - e o arquivo *targets.csv* contém tuplas do tipo {*user, item*} representando os alvos da predição. Espera-se como resposta um arquivo de nome *submission.csv* contendo UserID:ItemId:Prediction como cabeçalho na primeira linha e seguido de *n* linhas, sendo *n* é o número de tuplas do arquivo *targets.csv*, que seguem o formato do cabeçalho e representam, respectivamente, o identificador do usuário, o identificador do item, e a predição da nota feita pelo programa. Esse arquivo de saida deverá ser submetido na competição do Kaggle[1].

2 DESENVOLVIMENTO

Nesta seção será discutido como ocorreu a modelagem e o desenvolvimento da implementação do sistema.

2.1 Modelagem

O programa desenvolvido utiliza estrutura nomeada *Colecao* que foi criada para representar tanto itens quanto usuários, essa estrutura contém algumas informações básicas a respeito da instância junto com três estruturas mais complexas modeladas como: um vetor para armazenar tuplas {*item ou usuário, nota*} com os quais a instância tem relações, um vetor do tipo *int* para armazenar os índices dos elementos vizinhos, e um *map* do tipo <*string, double*> para mapear uma *string* a um valor de similaridade.

A parte principal do programa contém uma matriz de notas *ixj* - sendo *i* e *j*, respectivamente, o número de itens no histórico e o número de usuários no histórico-, um vetor de *Colecao* para armazenar os itens, um vetor de *Colecao* para representar os usuários e um *map* para mapear os identificadores de entrada a um inteiro que representa o índice do elemento no vetor de *Colecao*.

Ao iniciar o programa, ocorre a chamada da função *lerEntrada* que é a responsável por ler o arquivo *ratings.csv*. Durante esse processo, a função preenche os vetores de itens e usuários e também realiza o mapeamento do identificador textual para o identificador numérico. Após esse processo, as notas são adicionadas a matriz de notas e normalizadas.

Em seguida, ocorre a construção da vizinhança, que será utilizada acessar os vizinhos de forma indexada ao invés de percorrer toda a matriz de notas. Nesse processo, dois itens são considerados vizinhos se existe um usuário que consumiu ambos. Após isso, ocorre o calculo da similaridade entre os itens da base, e por fim é chamada a função *lerTargets* que é a responsável por gerar um vetor de tuplas a partir do arquivo *targets.csv* e ocorre a predição das notas.

2.2 Testes

O primeiro modelo testado para realizar a predição foi um modelo simples que utiliza médias para fazer a predição. Ele considera que o valor de um objeto é a média de suas notas menos a média global de notas ($valorObjeto = mediaObjeto - mediaGlobal$) e calcula a predição utilizando a fórmula $mediaGlobal + valorUsuario + valorItem$- caso o objeto não tenha sido visto, ele recebe o valor 0. A primeira implementação desse modelo atingiu um RMSE de 1.69 na competição do Kaggle e foi utilizada como base para o desenvolvimento do sistema. Após alguns ajuste de normalização, esse modelo consegui atingir um RMSE de 1.68.

Em seguida, foram implementados os algoritmos de calculo de similaridade de cosseno e o de coeficiente de correlação Pearson. Ambos foram utilizados no cenário onde o item e o usuário são conhecidos e, surpreendentemente, ambos obtiveram um resultado pior do que o modelo base. Ocorreram tentativas de balancear os índices de similaridade levando em consideração a quantidade de vizinhos do item, entretanto essas tentativas não resultaram em uma predição melhor que a do algoritmo base sendo 1.74 o melhor RMSE obtido por ambos.

Diante desse cenário, a decisão tomada foi a de mesclar as duas soluções. Sendo assim, caso o item e o usuário sejam conhecidos, ocorre o calculo da predição utilizando a similaridade de cosseno e o resultado final é a média entre essa predição e a predição feita pelo sistema base. Com esse método, a submissão feita no Kaggle obteve um RMSE de 1.67.

3 ANÁLISE DE COMPLEXIDADE

As funções que leem os arquivos de entrada não realizam cálculos complexos, sendo assim eles tem uma complexidade linear. Após a leitura e armazenamento dos dados, ocorre o preenchimento da

[1] https://www.kaggle.com/c/recsys-20191-cfmr/data

matriz de notas que, sendo N o número de itens e M o número de usuários, tem tamanho e complexidade O(NM).

O calculo dos itens vizinhos ocorre através da lista de relações de usuários. Como é necessário gerar todos os pares de itens vizinhos, ou seja, percorrer dois *loops* aninhados no pior caso - onde todos os usuários consumiram todos os itens- a função tem complexidade assintótica O(MN^2).

A última função a ser avaliada é a função que calcula a similaridade entre todos os itens. Ela precisa percorrer dois *loops* aninhados de tamanho N e, para cada uma dessas interações, percorrer a lista de relações que contém usuários consumidores. Sendo assim, o complexidade assintótica da função é O(MN^2).

Por fim, o programa percorre o vetor que contém todas as tuplas do arquivo *targets.csv* para gerar as predições, ou seja, tem complexidade linear. Sendo assim, pode-se concluir que a complexidade assintótica do programa é O(MN^2).

4 CONCLUSÃO

Nesse trabalho foi proposto a construção de um sistema de recomendação de filmes baseado em filtragem colaborativa. A resolução foi feita utilizando um sistema híbrido que utiliza uma função envolvendo as médias, tanto dos itens, quanto dos usuários, junto com a similaridade de cosseno para calcular as predições.