Documentação *Programming Assignment* 2

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

1 INTRODUÇÃO

O objetivo deste trabalho é implementar um recomendador de itens utilizando técnicas de recomendação baseadas em conteúdo. De modo geral, recomendações baseadas em conteúdo utilizam as semelhança entre as *features* do item sendo avaliado com as *features* dos itens que foram consumidos e/ou avaliados pelo usuário para realizar a predição.

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

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

onde o arquivo *content.csv* é o arquivo que contém as informações de conteúdo dos itens, *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 saída 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 chamada *Colecao* que foi criada para representar tanto itens quanto usuários, essa estrutura contém algumas informações básicas a respeito da instância junto com um vetor para armazenar tuplas {*item ou usuário, nota*} com os quais a instância tem relações.

A parte principal do programa contém um vetor de *Colecao* para armazenar os itens, um vetor de *Colecao* para representar os usuários, um *map* para ligar os identificadores de entrada ao inteiro que representa o índice do elemento no vetor de *Colecao*, um vetor do tipo *map<string, string>* para representar o conteúdo dos itens em sua forma bruta e um vetor de *string* para representar os itens.

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.

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

Em seguida, ocorre a chamada da função *lerConteudo* que é a responsável por ler o arquivo *content.csv*. Esse arquivo tem um formato semelhante ao de um arquivo JSON o que permite extrair os valores das *features* pela posição dos cabeçalhos. A função retorna um vetor, que está ordenado da mesma forma que o vetor de itens, com todas as informações de conteúdo. A partir desses dados, é construído um vetor de *string* para representar cada um dos itens.

Por fim, o programa lê o arquivo que contém as tuplas de *targets* e calcula a predição baseado no vetor de características.

2.2 Testes

O primeiro modelo testado para realizar a predição foi um modelo simples que realiza a predição baseado no compartilhamento de *features*. O modelo somava a nota de um item consumido pelo usuário para cada *feature* compartilhada entre o item do histórico e o item alvo. O resultado da predição era a média dos valores, ou seja, o somatório dividido pelo número de *features* compartilhadas. Na primeira submissão, o modelo atingiu um RMSE de 1.9.

Em seguida, foram implementadas funções para criar a representação TF-IDF de cada documento e para construir a matriz rocchie da base de dados. Entretanto, devido a um provável erro de implementação, os resultados não saíram como o esperado e, em sua maioria, obtiveram um RMSE acima de 2.5 . Após algumas alterações e testes, o modelo atingiu um RMSE de 2.1, resultado que é pior que o caso base.

Diante disso, a decisão tomada foi voltar para a implementação para a primeira versão e tentar aprimorar a representação dos itens. Após diversos testes, o melhor resultado obtido foi o que utiliza apenas os gêneros dos itens para realizar a predição. O modelo atingiu um RMSE de 1.8.

3 ANÁLISE DE COMPLEXIDADE

As funções que leem os arquivos de entrada *targets.csv* e *ratings.csv* não realizam cálculos complexos, sendo assim elas têm uma complexidade linear.

A função que faz a leitura do conteúdo quebra as linhas de entrada de acordo com o cabeçalho. Sendo I o número de itens do arquivo, T o tamanho máximo do conteúdo de cada item e C o tamanho máximo do cabeçalho($C \leq 10$), como a o trecho de código executa a função *string::find* 21 vezes para cada item e a função *find* tem complexidade O($T \times C$), o trecho tem complexidade O($I \times T \times C$) $\approx$ O($I \times T$) já que $I \gg C$ e $T \gg C$.

Após a leitura e armazenamento dos dados, ocorre o cálculo do vetor de representação dos itens sendo N o número de itens, como o trecho realiza comparação com a função *string::find*, o trecho apresenta complexidade assintótica O($I \times T$). Neste caso, T é limitado pelo comprimento máximo do campo *Genre* da base de dados.

Por fim, o programa percorre o vetor que contém todas as tuplas do arquivo *targets.csv* para gerar as predições utilizando a função *find* na representação dos itens,ou seja, cada predição tem custo

O(T), sendo T limitado pelo comprimento máximo do atributo *Genre* dentro base de dados. Nesse cenário, sendo M o número de tuplas do arquivo, o trecho tem complexidade assintótica O($M \times T$).

Por fim, é possível dizer que o programa tem um custo diretamente ligado ao tamanho da entrada. Dado que o arquivo *ratings.csv* tem N itens e o arquivo *targets.csv* tem M tuplas, o programa tem complexidade assintótica O($N + M \times T + I \times T$)

4 CONCLUSÃO

Nesse trabalho foi proposto a construção de um sistema de recomendação de itens baseado no conteúdo dos mesmo. A resolução foi feita utilizando um sistema que verifica a semelhança do item alvo com os itens consumidos pelo usuário, sendo que o valor da nota é o valor da média ponderada pela semelhança.