Relatório Trabalho Prático 1 – MovieLens

Rafael Glater da Cruz Machado

3 de setembro de 2014

1. Proposta

Implementação de um sistema de recomendação de filmes usando a abordagem baseada no usuário. Como base de dados de teste foi fornecido uma lista da MovieLens com ratings de usuários em diferentes filmes. A aplicação deve recomendar os top 100 filmes que ainda não foram avaliados pelo usuário em ordem decrescente de rating previsto.

2. Implementação

A aplicação foi desenvolvida em Java. A figura 1 ilustra em um diagrama simplificado as classes criadas e suas relações.

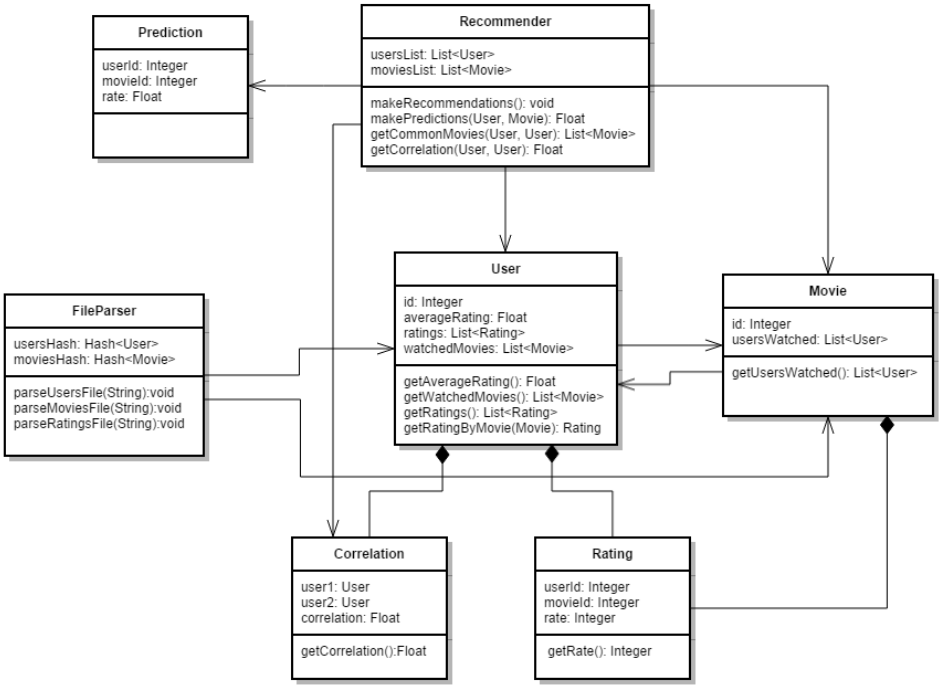

Figura 1. Diagrama de classes da aplicação

Foram criadas as seguintes classes para representar as entidades da aplicação: *User*, *Movie*, *Rating*, *Correlation* (para representar o valor da correlação de Pearson entre dois usuários) e *Prediction* (para representar a nota prevista de um usuário para um filme).

A classe *User* basicamente contém uma lista de *Movie* – para representar os filmes que o usuário já avaliou – e uma lista de *Rating* – para armazenar o valor da avaliação já feita em cada filme. A classe *Movie* contém uma lista de *User* para armazenar os usuários que avaliaram determinado filme.

A classe *FileParser* é responsável por varrer o conteúdo dos arquivos da MovieLens e criar as listas de usuários, filmes e ratings.

Após todos os dados estarem estruturados na memória, a classe *Recommender* será a responsável por fazer as predições das notas dos usuários. A classe *Recommender* possui, dentre outros, os seguintes métodos:

- *getCommonMovies(User, User)* – busca as notas de filmes em comum entre dois usuários;
- *getCorrelation(User, User)* – calcula a correlação de Pearson entre dois usuários;
- *makePrediction(User, Movie)* – calcula a nota prevista de um usuário para determinado filme;
- *makeRecommendations()* – faz a predição para todos os usuários, selecionando os top 100 filmes de cada um.

O método *makeRecommendations* varre todos os usuários e para cada um, busca os filmes que ainda não foram avaliados (fazendo a diferença da lista total de filmes com a lista de filmes do usuário). Para cada filme não avaliado pelo usuário é chamado o método *makePrediction* que implementa a fórmula da predição usando a correlação de Pearson ao chamar o método *getCorrelation*, este último chama o método *getCommonMovies* para fazer o cálculo de Pearson com os filmes em comum entre cada usuário.

O algoritmo abaixo mostra a implementação desses métodos e suas relações:

Método *makeRecommendations()*:

```
para cada user na usersList faça (N)
    newMovies = moviesList – user.watchedMovies
    predictionsList = nova lista

    para cada movie na newMovies faça (M)
        rate = makePrediction(user, movie)
        predictionsList.adiciona(user, movie, rate)
    fim para

    para cada prediction na predictionsList até 100 faça (100)
        escreva "user movie rate"
    fim para
fim para
```

A complexidade nesse caso é NM * makePredictions + 100N

Método *makePrediction(user, movie)*:

```
correlationsList = getCorrelationsList(user, movie)
numerador = 0
denominador = 0

para cada correlation na correlationsList faça (50)
    numerador += correlation * ( movieRateUser2 - averageUser2 )
    denominador += |correlation|
fim para

prediction = averageUser1 + numerador/denominador
retorna prediction
```

A média usada nesse método é a média global, levando em consideração todos os filmes que o usuário já avaliou. O loop desse método não passa de 50 para cada usuário, portanto a complexidade seria de 50N, e a média global é calculada apenas uma vez, portanto é N.

Método *getCorrelationList(user, movie)*:
Esse método separa os 50 vizinhos com a melhor correlação de Pearson.

```
usersWatched = movie.getUsersWatched()
correlationsList = nova lista

para cada user2 na usersWatched faça (N)
    correlation = getCorrelation(user, user2)
    correlationsList.adiciona()
fim para

ordena(correlationsList)
retorna sublista(correlationsList, 0, 50)
```

Para cada usuário N (do loop do método *makeRecommendations*) é percorrida a lista de usuários N para verificar pelos usuários que já viram o filme. Logo N^2.

Método *getCorrelation(user1, user2)*:

```
numerador = 0
denominador = 0
denominadorPart1 = 0
denominadorPart2 = 0

para cada rateUser1 e rateUser2 na commonMovies(user1, user2) faça
    numerador = (rateUser1 - averageUser1) * (rateUser2 - averageUser2)
    denominadorPart1 += (rateUser1 - averageUser1)^2
    denominadorPart2 += (rateUser2 - averageUser2)^2
fim para

denominador = raiz(denominadorPart1) * raiz(denominadorPart2)
correlation = numerador/denominador
retorna correlation
```

O método *commonMovies* faz uma interseção entre as listas de ratings em filmes do *user1* com o *user2*, portanto sua complexidade é de NM^2. A média usada nesse método é a média dos itens que são em comum entre os dois.

Juntando a complexidade dos métodos tenderia à $O(N^2M+NM^2)$.

3. Análise do resultado

Para analisar se a predição realizada pela aplicação estava próxima à real, foi calculada a predição de alguns dados já existentes. A Tabela 1 ilustra os dados obtidos.

Usuário	Filme	Nota real	Nota prevista	Vizinhos	> Pearson	< Pearson
1	225	2	2.79	50	0.99	0.22
1	213	2	3.63	50	0.99	0.24
1	219	1	3.18	50	0.99	0.30
1	216	1	3.89	50	0.99	0.37
1	199	4	4.35	50	0.99	0.30
1	240	3	2.88	50	0.99	0.33
1	253	5	4.08	21	0.99	0.42
2	100	5	4.62	50	1	0.69
2	255	4	3.48	50	0.99	0.31
2	305	3	3.82	50	0.99	0
2	315	1	3.83	50	0.99	0.23

Tabela 1. Comparativo entre dados reais e previstos

Estranhamente quando a nota real é baixa a aplicação prevê uma nota muito discrepante, mesmo encontrando o número máximo de vizinhos usados (50) e com valores de similaridade entre os vizinhos alto. Para os casos onde a nota real é alta, a aplicação consegue prever mais adequadamente.

Foi analisado também a saída dos filmes recomendados no top 100 de alguns usuários, representada na Tabela 2.

Usuário	Filme	Ranking	Nota Prevista	Vizinhos
1	1654	1°	6.14	1
1	814	2°	5.51	1
1	1039	99°	4.14	50
1	510	100°	4.14	50
2	1678	1°	5.82	1
2	814	2°	5.66	1
2	694	99°	4.29	38
2	251	100°	4.29	41
3	599	1°	5.86	1
3	1621	2°	5.71	1
3	498	99°	3.51	50
3	48	100°	3.51	50

Tabela 2. Análise dos dados previstos

Os filmes no topo do raking, que preveram notas inclusive maior do que 5, quase não possuem dados para serem comparados e previstos adequadamente. Como podemos notar, foram encontrados apenas um vizinho – com poucos filmes em comum. Já os valores mais ao final do ranking tiveram um valor possivelmente verossímil, pois foi levado em consideração o número de vizinhos adequados e com bastante filmes em comum.

A abordagem user-based é falha nesses pontos em que o usuário possui poucos dados para serem comparados com o restante da comunidade. Uma alternativa para isso seria colocar pesos

diferentes em usuários com poucos filmes em comum e levar em consideração também o número de vizinhos analisados. Uma outra alternativa poderia também usar uma abordagem híbrida, levando em consideração o conteúdo, como o estilo de filmes que o usuário já demonstrar ter preferência, por exemplo.