PA #1 - Collaborative Movie Recommendation

Ramon Gonçalves Gonze

16 de abril de 2018

Resumo

Este trabalho apresenta um recomendador de filmes colaborativo, isto é, que utiliza filtragem colaborativa. O objetivo é fazer predições de notas que usuários dariam para filmes que ainda não assistiram, através de recomendações personalizadas e não-personalizadas. Os resultados dos experimentos são apresentados ao final juntamente com a análise sobre a variação de parâmetros como o tamanho de k para a seleção de vizinhos, o tratamento de *cold-start*, entre outros, e seus respectivos impactos.

1 Introdução

A recomendação de filmes é amplamente utilizada nos dias atuais como forma de, por exemplo, aumentar vendas, como é o caso da Netflix. Existem várias abordagens para realizar tal objetivo, e este trabalho apresenta o *item-based*, que é a recomendação de itens (neste caso filmes) através de informações sobre outros itens que usuários já avaliaram no passado.

A base de dados utilizada contém várias notas de diversos usuários para diversos filmes que esses já assistiram. O modelo é feito em cima dessas notas já registradas, e o objetivo é realizar predições de notas que usuários dariam para filmes que não assistiram. A descrição dessa base é feita na Seção 3.

A modelagem do problema é descrita na Seção 2, os exeperimentos e suas respectivas análises se encontram na Seção 4 e por fim, a última seção conclui todo o trabalho realizado, exibindo as dificuldades encontadas e os resultados gerais.

2 Modelagem

A representação das relações entre usuários e filmes, ou seja, as notas, foi modelada em um grafo $G = (V, E)$ não-direcionado, bi-partido e ponderado, onde V é o conjunto de vértices e E é o conjunto de arestas. O conjunto de vértices pode ser particionado em dois conjuntos disjuntos U e F, sendo que U contém os vértices que representam os usuários e F contém os vértices que representam os filmes. O peso de uma aresta (u, f) (ou (f, u)) representa a nota que o usuário u deu para o filme f. Caso um usuário não tenha visto um filme, não haverá aresta eles. Cada vértice também possui um atributo *média*, que representa a média das notas dadas, no caso de um usuário, ou recebidas, no caso de um filme.

2.1 Similaridade entre filmes

Para o cálculo de similaridade entre filmes foi utilizado o cosseno dos vetores que representam os filmes. Como o problema foi modelado com grafos, o vetor de um filme é a sua lista de vértices adjacentes, ou seja, os usuários que o assistiram. A similaridade entre dois fimes f e g é dada por

$$sim(f, g) = \frac{\sum\limits_{u \in U} r_{uf} \cdot r_{ug}}{\sqrt{\sum\limits_{u \in U} r_{uf}^2} \cdot \sqrt{\sum\limits_{u \in U} r_{ug}^2}} \tag{1}$$

onde r_{uf} e r_{ug} são respectivamente as notas que u deu para f e g. O acesso às notas r_{uf} e r_{ug} são feitas em tempo $O(log|V|)$, através de uma estrutura de dicionário. O somatório é executado $|U|$ vezes, logo a a similaridade entre dois itens é calculada em $O(|U| \cdot log|V|)$.

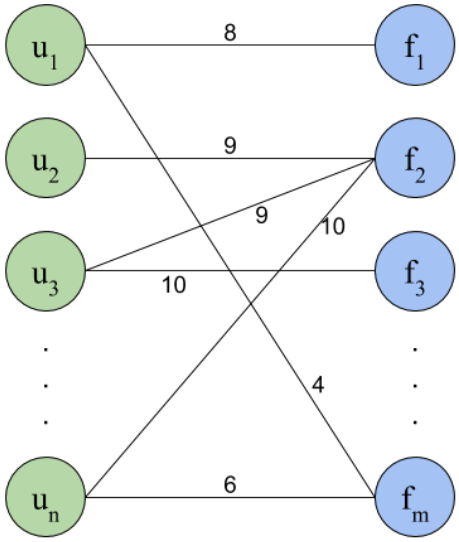

Figura 1: Representação das relações entre o conjunto de usuários $U = \{u_1, u_2, \ldots, u_n\}$ e o conjunto de filmes $F = \{f_1, f_2, \ldots, f_m\}$. Neste grafo, n e m são respectivamente o número de usuários e filmes.

2.2 Predição

Para predizer a nota de um usuário u para um filme f, deve-se cacular a similaridade entre todos os filmes que u já assistiu (e avaliou) e f. Após feito, se calcula a média ponderada das notas dadas por u a esses filmes, pesando a nota de cada filme g com sua respectiva similaridade $sim(f, g)$.

Uma técnica utilizada para melhorar a predição é a normalização. Usuários fornecem diferentes feedbacks para o sistema de recomendação. A variabilidade entre as notas de diferentes usuários pode variar bastante. Por exemplo, existem usuários que avaliam filmes somente com notas altas, enquanto outros podem avaliar filmes somente com notas baixas. A normalização dessas notas busca ignorar essas diferenças dessa variabilidade. Uma das formas de normalização é o *mean-centering*, que consiste em, no momento de uma predição, subtrair a média das notas recebidas pelo filme. Ao final, soma-se a média das notas desse filme ao resultado obtido.

Seja $p(u, f)$ a predição da nota do usuário u para o filme f e A o conjunto de filmes assistidos por u,

$$p(u, f) = \bar{r}_f + \frac{\sum_{g \in A} sim(f, g) \times (r_{ug} - \bar{r}_g)}{\sum_{g \in A} sim(f, g)} \tag{2}$$

onde r_{ug} é a nota que o usuário u deu para o filme g, $\bar{r}_f$ a média das notas recebidas pelo filme f e $\bar{r}_g$ a médias das notas recebidas pelo filme g. O somário em (2) é executado $|A|$ vezes. No pior caso, o usuário u já avaliou todos os filmes, então o somatório é executado $|F|$ vezes. Sabendo a complexidade de $sim(f, g)$ (demonstrada na seção anterior), a complexidade de $p(u, f)$ será $O(|F| \cdot |U| \cdot log|V|)$.

2.3 Cold-Start

Um dos problemas enfrentados por recomendadores de filmes é o *cold-start*, que acontece quando se tem pouca ou nenhuma informação sobre usuários e/ou filmes. A base de dados adotada não possui informações como idade, gênero e localização dos usuários e nem informações como diretor, atores e gênero dos filmes. Portanto, técnicas como *content-based* para tratar o cold-start não puderam ser adotadas. As estratégias para tratar este problema são descritas a seguir.

Seja `<u:f>` o par usuário/item no qual queremos predizer a nota de u para f. Há quatro situações possíveis:

1. u já avaliou algum filme e f já foi avaliado por alguém.

2. u já avaliou algum filme e f não foi avaliado por ninguém.

3. u não avaliou nenhum filme e f já foi avaliado por alguém.

4. u não avaliou nenhum filme e f não foi avaliado por ninguém.

As seguintes medidas foram adotadas para as respectivas situações:

1. Calcula-se o valor da predição conforme em (2);

2. Retorna-se a média das notas de u;

3. Retorna-se a média das notas de f;

4. Retorna-se a nota 7.0.

O experimento da Seção 4.2 avalia o melhor valor a ser retornado para a situação 4.

2.4 Seleção de vizinhos

Dada uma predição $p(u, f)$ a ser realizada, a seleção dos k vizinhos mais próximos se consiste em escolher os k filmes mais similares a f, ou seja, que possuem maiores valores para $sim(f, g)$, para todo $g \in F$. Logo, k representa o valor de $|A|$ em (2). O valor de k não pode ser muito alto, pois introduz muito ruído, e nem muito baixo, pois dá pouca cobertura para o cálculo da similaridade. O experimento da Seção 4.1 indica o melhor valor encontrado para k.

2.5 Avaliação

A métrica utilizada para avaliação é a Raiz Quadrada do Erro Quadrático Médio (RMSE). Ela retorna o erro médio da diferença entre a nota real do usuário para um filme e a predição da nota feita pelo recomendador, para todo par `<usuário:item>` $\in C_2$ (vide Seção 3). Seja $r(u, f)$ o valor real que o usuário u deu para o filme f,

$$RMSE = \sqrt{\frac{1}{n} \sum_{(u,f) \in C_2} (r(u, f) - p(u, f))^2} \tag{3}$$

onde $n = |C_2|$.

2.6 Análise de Complexidade

A complexidade temporal do programa principal se baseia na complexidade de $p(u, f)$ demonstrada na Seção 2.2. Primeiramente é feita a leitura das arestas do grafo (as notas) e construído o grafo. Gasta-se $O(log|V|)$ para se inserir uma aresta. Logo, para construir o grafo, gasta-se $O(|E| \cdot log|V|)$. Seja P o número de pares (u, f) do conjunto C_2 (Seção 3). Para cada predição, o programa principal chama a função $p(u, f)$. O programa salva na memória sempre que calcula uma similaridade entre dois filmes. Contudo, no pior caso todos os pares (u_i, f_i) contém filmes f_i nos quais não tiveram sua similaridade calculada com nenhum filme visto por u_i.

Ao final tem-se a complexidade como $O(|E| \cdot log|V|) + O(P \cdot |F| \cdot |U| \cdot log|V|)$. Como $|U|$ e $|F|$ são subconjuntos disjuntos de $|V|$, e $|U| \cup |F| = |V|$, então $|U| \cdot |F| \geq |V|$ (considerando que nenhum conjunto é vazio). Logo a complexidade do programa principal será $O(P \cdot |F| \cdot |U| \cdot log|V|)$.

3 Base de dados e Execução

A base de dados está dividida em dois conjuntos C_1 e C_2, onde C_1 é o conjunto que contém as notas que filmes receberam de usuários no passado, no formato `<usuário:filme,nota,timestamp>`[1] , e este contém 336.672 notas. As notas são números inteiros no intervalo $[0, 10]$. Já o conjunto C_2 contém uma lista de 77.276 pares `<usuário:item>`, os quais são utilizados para serem feitas as predições.

[1] A base utilizada possui informação sobre o *timestamp*, porém esse não foi utilizado no recomendador.

O trabalho foi implementado em C++, utilizando a biblioteca padrão. Há um arquivo principal com o nome *recommender* que possui a função principal do programa, e mais dois módulos *prediction* e *graph*, que contém, respectivamente, as funções que calculam (1) e (2) e as funções necessárias para implementar o grafo descrito na Seção 1. O algoritmo recebe dois parâmetros por linha de comando, sendo o primeiro o nome do arquivo que contém C_1 e o segundo o arquivo que contém C_2.

Há um *Makefile* que compila o programa, e gera um executável `recommender`. Portanto, deve-se executar o comando `make` no terminal e posteriormente

```
$./recommender <arquivo_C1> <arquivo_C2>
```

Exemplo de execução do programa:

```
$./recommender ratings.csv targets.csv
```

A saída do programa é feita pela saída padrão (**stdout**).

4 Experimentos

Os experimentos foram realizados com o intuito de encontrar os melhores valores para k (Seção 2.4) e para a situação 4 (Seção 2.3), além de testar outras técnicas (Experimento 3). Os testes exaustivos foram realizados em um ambiente Linux, utilizando a distribuição Ubuntu 16.04 LTS.

4.1 Experimento 1 - k-vizinhos

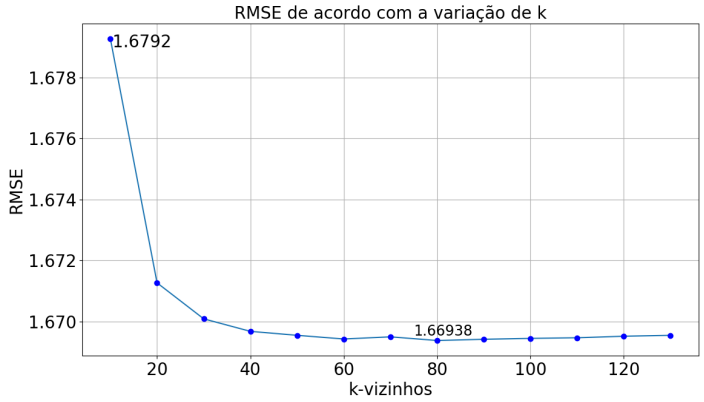

Figura 2: Impacto da seleção de diferentes valores para k no RMSE.

Pode-se observar na Figura 2 que o valor de k não é um fator determinante no desempenho do recomendador. O RMSE varia em, no máximo, 0,01. O menor erro encontrado com o recomendador foi de **1.66938**, quando $k = 80$.

4.2 Experimento 2 - Cold-start

Este experimento avalia o melhor valor para se retornar na situação 4 descrita na Seção 2.4.

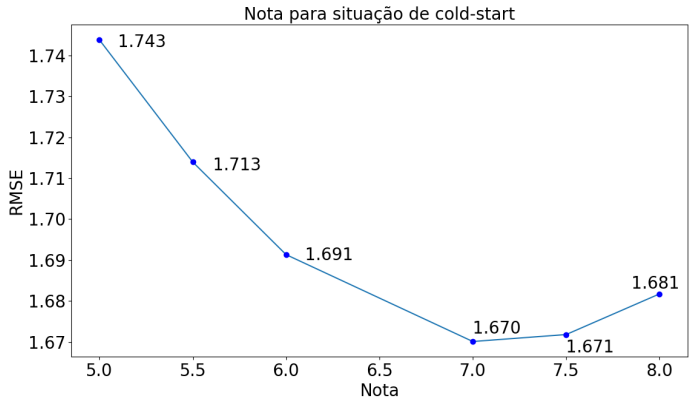

Figura 3: Impacto do valor da nota retornada para uma situação de cold-start no RMSE.

De acordo com o gráfico da Figura 3, pode-se observar que valores acima e abaixo de 7.0 alcançam piores RMSE. Foi adotado então o valor 7.0 para se retornar na situação 4.

4.3 Experimento 3 - Ponderamento de similaridades

Uma outra possível abordagem a ser utiliza é a de ponderação das similaridades caculadas em (1). Ela consistem em basicamente determinar um valor de confiança c no qual o valor de uma similaridade $sim(f, g)$ é multiplicado por $min(n, c)/c$, onde n é a quantidade de usuários que avaliaram tanto f quanto g.

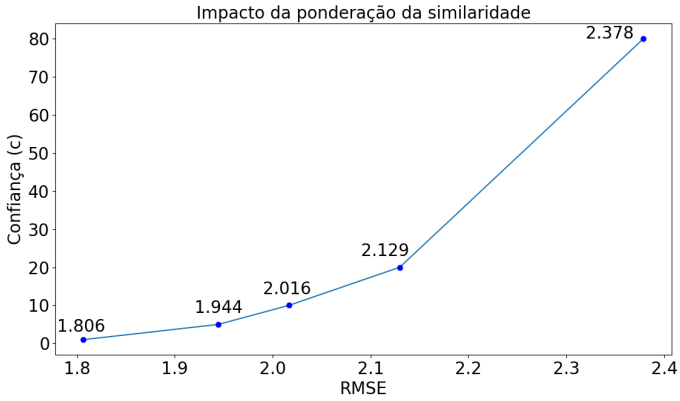

Figura 4: Impacto do valor de c no desempenho do recomendador.

Através da Figura 4 concluímos que a abordagem de ponderação das similaridades não é adequada para este contexto. Logo, optou-se por não utilizá-la.

5 Conclusão

A partir dos resultados descritos no corpo deste trabalho, pode-se concluir que a abordagem *item-based* é interessante para ser utilizada no domínio de filmes. A similaridade entre dois filmes com o cosseno apresentou-se como um cálculo eficaz, mas em contrapartida, ponderar as similaridades pela confiança se mostrou ineficaz. Era esperado que, com a ponderação de similaridades, melhores resultados fossem alcançados, porém o Experimento 3 mostrou o contrário (ao menos para esta base de dados e para este algoritmo).

Também era esperado que o valor dos k-vizinhos influenciasse de forma considerável o desempenho do recomendador, porém, empiricamente foi demonstrado o contrário.

As estruturas de dados disponíveis na biblioteca padrão de C++ se mostraram bastante eficientes para a execução do algoritmo, como por exemplo, a estrutura *map* para a construção do grafo.

Referências

[1] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.