RecSys - Trabalho Prático 1 - Filtragem Colaborativa

Lucas de Miranda Bastos
Departamento de Ciência da Computação
Instituto de Ciências Exatas
Universidade Federal de Minas Gerais
lucasmbastos@dcc.ufmg.br

1. INTRODUÇÃO

Este documento descreve a resolução do trabalho prático 1 da disciplina de Sistemas de Recomendação do Departamento de Ciência da Computação da Universidade Federal de Minas Gerais. O trabalho consiste em desenvolver um sistema de filtragem colaborativa para um *dataset* de avaliação de filmes.

A implementação da solução é feita utilizando técnicas de filtragem colaborativa baseada em item, como visto em sala de aula. Essa abordagem foi escolhida por demonstrar melhor eficiência, eficácia e estabilidade ao ser comparada com abordagens de sistemas colaborativos baseados em usuário.

Nesta abordagem as avaliações são modeladas em uma matriz R de duas dimensões (colunas x itens) e cada célula armazena a avaliação de um item feita por um usuário. Dessa forma um item pode ser representado por um vetor $\vec{i}$ em um espaço $\mathbb{R}^N$ onde N é o número de usuários conhecidos.

Em posse dessa matriz R, a predição é feita com base em várias operações vetoriais que serão descritas em detalhes na seção 2. Existem diversas variações que podem ser feitas em várias etapas do algoritmo e não existe uma receita para qual é a melhor a ser usada, por isso, em áreas como sistemas de recomendação ou aprendizado de máquina, é comum fazer testes alterando os hiperparâmetros do algoritmo.

Nas seções seguintes serão descritas as decisões de projeto e implementação (seção 2), as análises e experimentações feitas para analisar o trabalho (seção 3,) e as considerações finais sobre o trabalho (seção 4).

2. ARQUITETURA E IMPLEMENTAÇÃO

O algoritmo de filtragem colaborativa baseado em item possui 4 passos fundamentais, cada passo possui uma série de variações de técnicas que podem ser utilizadas para melhorar os resultados de predições. A seguir, serão enumerados quais são esses passos e quais as abordagens utilizadas na versão final do programa.

- **Normalização**. A normalização é uma técnica muito importante para melhorar resultados de recomendação ao retirar os vieses de avaliações dos itens e dos usuários. Para o trabalho foi utilizada a técnica de *mean centering*. Dada uma avaliação $r_{i,j}$ feita por um usuário i para um item j, o novo valor $r_{i,j}{}'$ é calculado como: $r_{i,j}{}' = r_{i,j} - \bar{r_i} - \bar{r_j}$, onde $\bar{r_i}$ e $\bar{r_j}$ são respectivamente as médias das avaliações dos usuários e as avaliações dos itens.

  Vale ressaltar que ao fazer essa normalização, o resultado da predição deve adicionar as média do usuário e do item.

- **Cálculo de Similaridade**. Uma importante etapa do algoritmo de filtragem colaborativa baseada em item consiste em encontrar itens que sejam parecidos com os item que se deseja avaliar. Como dito na seção 1 os itens são modelados como vetores no espaço dos usuários, por isso a similaridade pode ser medida através de alguma técnica de cálculo de similaridade de vetores. A técnica escolhida para a versão final do trabalho é a similaridade do cosseno. $sim(\vec{a}, \vec{b}) = cos(\vec{a}, \vec{b}) = \frac{\sum_i^N r_{i,a} * r_{i,b}}{\sqrt[2]{\sum_i^N r_{i,a}^2} * \sqrt[2]{\sum_i^N r_{i,b}^2}}$.

- **Número de Vizinhos**. O trabalho considera todos os vizinhos do item ao fazer a predição. Essa decisão adiciona bastante ruído na predição e poderia ser otimizada. Entretanto, devido a má gestão de tempo do autor, não foi possível adicionar melhora em tempo de submissão para o *Kaggle*.

- **Predições**. Por fim, a predição $\hat{r}_{i,j}$ é feita pela média das avaliações dos outros itens consumidos pelo usuário i poderada pela similaridade desses itens com o item j. $\hat{r}_{i,j} = \frac{\sum_n^N (sim(\vec{j}, \vec{n}) * r_{i,n})}{\sum_n^N sim(\vec{j}, \vec{n})}$

- **Predições onde não se conhece todas as informações**. Podem existir casos onde a predição envolve um item e/ou um usuário não conhecido. Nesses casos, são adotadas algumas predições especiais.

  - *Não se conhece o usuário.* Utiliza-se a média do item a ser predito.
  - *Não se conhece o item.* Utiliza-se a média do usuário.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

– *Não se conhece nem o usuário nem o item.* Utiliza-se uma média geral dos itens, isso é, a média das médias dos itens.

A linguagem utilizada para a realização do trabalho foi a linguagem *C++* com a *standart library* versão 11 que é muito conhecida pela sua eficiência e praticidade em relação a versões antigas e linguagens como *C*. Como não foi permitida a utilização de bibliotecas de terceiros, as implementações foram feitas utilizando estruturas nativas do do *C++*.
A matriz R foi implementada utilizando um *hashmap* de *hashmap* (`std::map<int, std::map<int, double> >`), dessa forma, indexar cada avaliação da seguinte maneira `R[id_i][id_u]`, onde `id_u` é o identificador do usuário e `id_i` é o identificador do item. A utilização de *hashmaps* é possível pois os *ids* podem ser convertidos para números inteiros e são únicos. Além disso, a escolha também é interessante pois a indexação é feita sobre uma árvore binária balanceada (*red-and-black tree*) onde todas as operações (inserção, busca e remoção) são feitas em $O(log(n))$. Como a maioria das operações incluem acessos à matriz, é possível recuperar qualquer item em complexidade logarítmica.
Devido a restrição de limite de tempo de execução, a otimização do mesmo é mais prioritária do que a otimização de memória e para isso foram criadas estruturas auxiliares para o problema, como *hashmaps* ou *sets* que armazenam médias e normas de cosseno. Como cada uma dessas grandezas é indexada a um usuário ou item, a escolha da estrutura de dados fornece a mesma vantagem supracitada.
Não é uma prática comum de sistemas de filtragem colaborativa realizar o cálculo de similaridade *on-line*, isso é, em tempo de execução. Normalmente as similaridades são calculadas previamente e armazenadas em memória permanente *off-line*, essa abordagem é preferível em relação a primeira pois evita fazer o mesmo cálculo várias vezes e é possível devida a alta estabilidade da matriz de itens. Porém, como o trabalho possui uma única execução, o autor achou preferível fazer os cálculos de similaridade *off-line*, pois para uma única execução, essa abordagem foi mais rápida do que realizar os cálculos *on-line*.

3. EXPERIMENTOS E ANÁLISES

3.1 Análise teórica

Para a realização da análise teórica, considere n, m o número total de usuários e itens respectivamente. Também considere r o número de avaliações a serem preditas.
A leitura dos dados do arquivo ocorre através de um laço que percorre o *dataset*, o número de iterações do laço é proporcional ao número de usuários e itens, a leitura é da ordem de $O(m*n)$. Ainda na leitura, os identificadores são normalizados e transformados em inteiros, a complexidade desse passo é proporcional ao número de caracteres dos itens e dos usuários, porém, como eles possuem no máximo 8 caracteres, o processo tem complexidade constante ($O(1)$). Por fim, a leitura envolve acessos e inserções aos vários *hashmaps* utilizados $O(log(m)+log(n))$. Portanto, a complexidade final da leitura é $\boldsymbol{O(m*n*(log(m)+log(n)))}$.

Os cálculos de norma de cosseno e as médias fazem um acesso e uma inserção aos *hashmaps* de item ou usuário m ou n vezes respectivamente, dessa forma a complexidade dessas etapas são:

- **Média dos usuários.** $O(n*log(n))$
- **Média dos itens.** $O(m*log(m))$
- **Norma dos vetores de itens.** $O(m*log(m))$
- **Normalização da matriz.** $O(n*m*(log(n)+log(m)))$

Por fim a predição é feita para r entradas utilizando um laço. Cada predição envolve selecionar todos os itens avaliados pelo usuário-alvo $O(m)$ e todos os usuários que avaliaram o item-alvo $O(n)$, o que daria uma complexidade final de $O(r*m*n)$, se $r \propto n$, então a complexidade é $O(r*m*n^2)$. Entretanto, essa análise pode ser amortizada uma vez que a matriz R é em geral muito esparsa. Ao selecionar todos os usuários que avaliaram um determinado item que foi avaliado pelo usuário-alvo, desperdiça-se processamento pois a maior parte das multiplicações no cálculo do produto escalar tem resultado 0. Dessa forma, é possível utilizar a operação de interseção de conjuntos para só considerar usuários comuns ao item-alvo e o item que está sendo avaliado. Dessa forma, É feita uma número k de multiplicações, onde $k << n$. Como a interseção é da ordem de $k*log(k)$ a complexidade da predição amortizada é $O(r*k*log(k)*n)$

3.2 Análise experimental

Os testes foram realizados nas máquinas disponíveis aos alunos do Departamento de Ciência da Computação pelo Centro de Recursos Computacionais. Os computadores possuem a seguinte especificação:
Memória de 16GB, Processador *Intel Core i7* sexta geração com *clock* de 3.40GHz. O sistema operacional é a distribuição do *GNU/Linux Ubuntu* 16.04. O compilador utilizado é o *g++*.
O tempo de execução do código para gerar a submissão pro *Kaggle* foi de 2 minutos e 36.316 segundos (medido com o utilitário `time`). Como o programa possui uma série de *loops*, principalmente na etapa de predição, foram adicionadas *flags* de otimização no comando de compilação. Executando o código sem a *flag* de otimização, o tempo de execução da mesma entrada é de 5 minutos e 34.496 segundos.

4. CONCLUSÕES

Desenvolver um sistema de recomendação do 0 é uma tarefa que envolve vários problemas práticos. O maior problema encontrado pelo autor ao desenvolver o trabalho foi relacionado com o acesso em matriz, uma vez que ao utilizar dois *hashmaps*, acessar uma linha é $O(n)$, mas o acesso a coluna não é trivial e envolve percorrer todas linhas, resultando em uma operação $O(m*n)$. Dessa forma, foram necessário o uso de várias variáveis auxiliares. Em linguagens que suportam operações matriciais, como *Matlab* ou *Octave*, essas operações não requerem estruturas auxiliares e são otimizadas para serem mais eficientes do que $O(n)$.
Um outro problema está relacionado com o formato de entrada e saída dos dados. O formato `.csv` é puramente um

formato de texto e as informações são separadas por vários delimitadores (no *datset* do trabalho foram utilizados 2 delimitadores). Dessa forma, parte do processamento envolve o *parsing* dessas informações que é praticamente feito de maneira artesanal, onde qualquer mudança na estrutura dos dados envolve mudança na lógica do código.
Por fim, o tralho foi muito proveitoso, entretanto poderia ter melhores resultados se o autor não tivesse gerenciado tão mal o tempo disponível para a execução do mesmo. Espera-se que para o próximo trabalho esse problema não aconteça novamente.