Lucivan Batista

Trabalho 2
Sistema de Recomendação
Filtragem Colaborativa

Índice

Introdução

- Filtragem Colaborativa
- Exemplo
- GroupLens
- MovieLens

Filtragem Colaborativa

- **Filtragem colaborativa** consiste na recomendação de itens que pessoas com gosto semelhante preferiram-no passado.
- Analisa-se a vizinhança do usuário a partir da regra: "Se um usuário gostou de A e de B, um outro usuário que gostou de A também pode gostar de B".
- Esse tipo de recomendação apresenta resultados positivos na prática, e evita o problema de recomendações repetitivas.
- Uma desvantagem é que requer grande número de informações sobre o usuário e sua vizinhança para funcionar precisamente.

Problemas da Filtragem Colaborativa

Problema do Primeiro Avaliador

- Necessidade de mais usuários para recomendar esse item para outros usuários.

Similaridade

- Encontrar uma similaridade para um usuário totalmente fora dos padrões.

Novo item

- Quando um novo item aparece e não faz parte do perfil de nenhum usuário, não é possível realizar recomendações ou predições sobre ele. (Pelo menos não com a filtragem colaborativa)

Novo usuário (*Cold Start User*)

- Não é possível encontrar vizinhos próximos, pois não há avaliações feita por ele.

Escalabilidade

- Quanto mais itens e mais usuários maior o cálculo dos vizinhos, de forma online fica bastante inviável o tempo de resposta.

Problemas da Filtragem Colaborativa

Esparsialidade

- Na medida que o número de itens na base de dados vai aumentando, isso reduz as chances dos usuários possuírem itens em comum, resultando na redução do tamanho médio da vizinhança dos usuários.
- Caso o número de usuários seja pequeno em relação ao volume de informações no sistema existe um grande risco das pontuações tornarem-se muito esparsas.
- Por conseguinte, o sistema terá menos vizinhos para realizar as predições, causando um impacto negativo na confiabilidade das recomendações.

Superespecialização

- Este problema refere-se ao fato do sistema só conseguir recomendar itens muito semelhantes àqueles que o usuário já avaliou.
- Por exemplo, um usuário que tem o perfil formado basicamente de filmes de guerra, receberá recomendações, em grande parte, de outros filmes de guerra.
- Isto pode tornar-se um problema, uma vez que os interesses dos usuários tendem a apresentar mudanças com o passar do tempo.

Exemplo

Exemplo

Usuário/ Item	A	B	C	D	E
U1	3		4	3	2
U2	4	5	4	4	2
U3	5	1	1	5	4
U4	4	5	3	3	3
U5	1	3		2	

GroupLens

- O GroupLens é um laboratório de pesquisa no Departamento de Ciências da Computação e Engenharia da Universidade de Minnesota, Twin Cities, especializado em sistemas de recomendação, comunidades on-line, tecnologias móveis e onipresentes, bibliotecas digitais e sistemas de informação geográfica local.

MovieLens

- A GroupLens Research coletou e disponibilizou conjuntos de dados de classificação disponíveis no site MovieLens.
- Os conjuntos de dados foram coletados em vários períodos de tempo, dependendo do tamanho do conjunto.
- Esses conjuntos de dados de classificação são relacionados a filmes e suas notas.

Dataset

- Arquivos CSV
  - Genome-Scores
  - Genome-Tags
  - Links
  - Tags
  - Movies
  - Ratings

Dataset

- Arquivos CSV
  - Genome-Scores
  - Genome-Tags
  - Links
  - Tags
  - **Movies**
  - **Ratings**

Movies.csv

- As informações do filme estão contidas no arquivo ‘movies.csv’. Cada linha deste arquivo após a linha do cabeçalho representa um filme e possui o seguinte formato:
  - *movieId, title, genres*
- Os títulos dos filmes são inseridos manualmente ou importados de <https://www.themoviedb.org/>, e incluem o ano de lançamento entre parênteses. Podem existir erros e inconsistências nesses títulos.
- Apenas os filmes com pelo menos uma classificação ou tag estão incluídos no conjunto de dados.
- Esses IDs de filme são consistentes com aqueles usados no site do MovieLens.
- Exemplo: id ‘1’ corresponde ao URL https://movielens.org/movies/1.
- As ID do filme são consistentes entre ‘ratings.csv’.
- Atributos: *movieId, title, genres*
- Quantidade de Tuplas: 45843

Ratings.csv

- Todas as classificações estão contidas no arquivo ‘ratings.csv’.
- Cada linha deste arquivo após a linha do cabeçalho representa uma classificação de um filme por um usuário e possui o seguinte formato:
  - userId, movieId, rating, timestamp
- As linhas deste arquivo são ordenadas primeiro pelo userId, então, dentro do usuário, pelo movieId.
- As classificações são feitas em uma escala de 5 estrelas, com incrementos de meia estrela.
  - (0,5 estrelas - 5,0 estrelas)
- Atributos: *userId, movieId, rating, timestamp*
- Quantidade de Tuplas: 26024289

Pré-processamento

1. Escolha e eliminação dos .csv desnecessários para a Filtragem Colaborativa
    1. Movies.csv
    2. Ratings.csv
2. Pentaho -> PostgreSQL
    1. Tabela Movies corresponde ao Movies.csv
    2. Tabela Ratings_new corresponde ao Ratings.csv
3. Eliminação de atributos desnecessários
    1. Em Movies foi removido o atributo *genres*
    2. Em Ratings foi removido o atributo *timestamp*
4. Normalização (ratings_norm)
    1. Criado uma function no PostgreSQL que recebe todas as tuplas e normaliza os dados
    2. Ratings = Ratings * 0,2
    3. Normalizar os dados para que fiquem entre 0 e 1

CREATE or REPLACE FUNCTION normalizar3() RETURNS void AS $$

DECLARE

- cursor1 CURSOR is select userid, movieid,rating,id from ratings_new;
- cont_id int; -- Contador para o loop
- userid ratings_new.userid%TYPE;
- movieid ratings_new.movieid%TYPE;
- rating ratings_new.rating%TYPE;

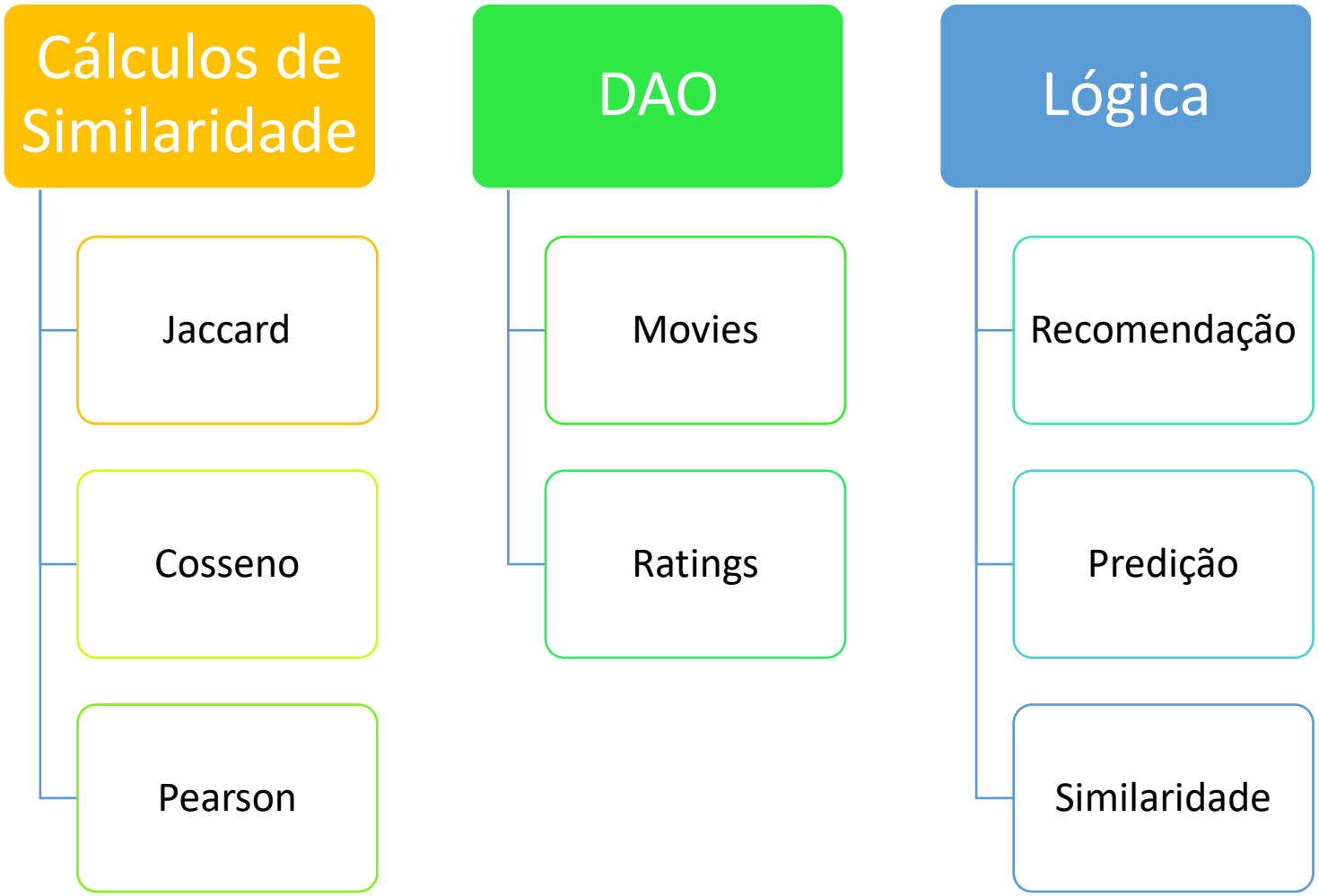
BEGIN

- OPEN cursor1; -- Cursor para o loop
- FETCH cursor1 into userid, movieid, rating;
- LOOP
  - EXIT WHEN NOT FOUND;
  - INSERT into ratings_norm values (userid, movieid,rating * 0.2);
  - FETCH cursor1 into userid, movieid, rating; -- Pula para o próximo no cursor
- END LOOP;
- CLOSE cursor1;

END;

$$ language plpgsql;

Function para Normalização no PostgreSQL



Filtragem Colaborativa – Programação

Parâmetros Utilizados

userIdA – ID do usuário escolhido

Divisor = 2 – Usado para pegar a metade dos usuários, mas pode ser alterar dentro do programa. Foi escolhido 2, pois é mais fácil pegar usuários similares

similarityPearsonP = 0.65 e similarityPearsonN = -0.65 – Usados para pegar os usuários mais similares, eles precisam ter 65% de similaridade.

qtdMinimunUserSimilar = 10 – Quantidade de Usuário similares necessários para realizar a predição, caso não consiga, o divisor é aumentado até encontrar uma quantidade igual ou superior a essa.

ratingMinimun = 0.7 – Usado para pegar os filmes pós predição que possuem nota >= 0.7 (3.5)

Otimização

- Buscar realizar um pré-processamento com os dados no PostgreSQL, buscando reduzir a quantidade de tuplas.
  - Exemplo no código: Pegar os filmes do usuário 1, para todos os ratings, pegar os usuários que possuem pelo menos a metade de quantidade de filmes que 1 deu rating (13) e salvar em uma outra tabela. (ratings_new_user)
  - De 26 milhões foi reduzido para 5 milhões.
  - Caso não encontre pelo menos 10 usuários semelhantes, a busca é reduzida para 1/3 dos ratings (9)

  - Exemplo: Dos 27 filmes que usuário 1 deu ratings, é melhor
  - (I) Pegar um usuário X que deu 3 ratings para os mesmos filmes que usuário 1, possuindo uma similaridade 1 ou -1.
  - (II) Pegar um usuário Y que deu 13 ratings para os mesmos filmes que usuário 1, possuindo uma similaridade acima de 0.7 ou abaixo de -0.7.

int qtdRatingsUser = this.getQtdRatingsUser(user) / divisor;

insert into ratings_new_user

- select * from ratings_norm where userid in
  - (select userid from ratings_norm where movieid in
    - (select movieid from ratings_norm where userid = " + user + ")
  - group by userid having count(userid) > " + qtdRatingsUser + ")”

Código insertTableReduction

Processamento de Otimização

01

Após a otimização de cortar usuários que não obteriam uma similaridade alta, todas as tuplas foram carregadas na memória. (List<Rating> ratings)

02

Depois foi criado um Map<Integer, List<Rating>>, a chave é um id de usuário e o valor é uma lista de Ratings que esse usuário deu. (ratingsAll)

- **for(Rating r : ratings){**
  - List<Rating> temp = ratingsAll.get(r.getUserId());
  - temp.add(r);
  - ratingsAll.put(r.getUserId(), temp);
- }

Para cada usuário possivelmente similar, será feito a similaridade com os ratings do Usuário escolhido

Caso esse usuário possua uma similaridade, utilizando-se a Pearson Correlation, maior que 0.7 ou menor que -0.7, então ele será considerado um usuário similar.

Observação: O garbage collector é sempre chamado quando uma variável não é mais usada, para liberar memória

Após encontrar os usuários similares, os filmes desses usuários são contados (Map<Integer, Integer> movies, key = id movie e value = cont), depois são retirados os filmes que o Usuário escolhido já assistiu e são removidos. Eu iria usar essa contagem para retirar filmes que poucas pessoas assistiram, mas isso iria prejudicar um pouco na predição e poderia remover um possível filme recomendado.

Resumindo os próximos passos

Pearson Correlation

$$f(a,u,I) = \frac{\sum_{i \in I}(r_{a,i} - \overline{r_a})(r_{u,i} - \overline{r_u})}{\sqrt{\sum_{i \in I}(r_{a,i} - \overline{r_a})^2 \sum_{i \in I}(r_{u,i} - \overline{r_u})^2}}$$

Predição

- São realizadas 2 predições
- Predição 1: “Média” dos ratings de cada usuário que avaliou aquele filme

$$P_{u,i} = \frac{1}{|\Gamma_u|} \sum_{v \in \Gamma_u} r_{v,i}$$

Predição

- Predição 2:

$$p_{a,i} = \overline{r_a} + \frac{\sum_{u \in K} (r_{u,i} - \overline{r_u}) \times w_{a,u}}{\sum_{u \in K} w_{a,u}}$$

Conclusões e Resultados

Referências

- https://grouplens.org/datasets/movielens/
- https://en.wikipedia.org/wiki/Collaborative_filtering
- http://movielens.org
- F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19. <https://doi.org/10.1145/2827872>
- https://www.infoq.com/br/presentations/algoritmos-de-filtragem-colaborativa-e-recomendacoes
- https://www.ibm.com/developerworks/br/local/data/sistemas_recomendacao/index.html
- Sampaio, Igor Azevedo. "Aprendizagem ativa em sistemas de filtragem colaborativa." (2006).
- Prof. Dr. Sílvio César Cazella. Slide Sistemas de Recomendação.