Ruído Natural na Filtragem Colaborativa

[Trabalho Final]

Paulo Xavier
contato.pauloxavier@gmail.com

Gabriel Segobia
segobia.gos@gmail.com

Fellipe Bravo
fellipe.bravo@gmail.com

ABSTRACT

Foi mostrado recentemente que os usuários podem ser inconsistentes quando eles elicitam avaliações para itens, expondo os dados usados nos SR à inconsistências. Esse trabalho busca quantificar o impacto do ruído na predição por modelos, e o desempenho dos algoritmos de detecção de ruído propostos por *O'Mahony* e *Toledo*.

1. INTRODUÇÃO

Sistemas de Recomendações(SR) são ferramentas de software e técnicas que fornecem sugestões de itens que sejam de uso ao usuário. Existem diversas abordagens que podem ser usadas em SR, e uma das mais populares é a *Collaborative Filtering(CF)*, que produz recomendações específicas a um usuário baseado em padrões de avaliação ou uso de itens.

Em SR é assumido que as avaliações nos *datasets* estão livres de irregularidades. Foi mostrado recentemente que os usuários podem ser inconsistentes quando eles elicitam avaliações para itens, expondo os dados usados nos SR à inconsistências. Esse tipo de inconsistência é conhecida como *ruído natural*. Abordagens que buscam *detectar* e *corrigir* essas avaliações inconsistentes surgiram para resolver esse problema em aberto.

O artigo será organizado da seguinte forma: Seção 3 faz uma revisão de filtragem colaborativa. Seção 4 elicita o problema e faz uma proposta para quantificar o impacto do ruído nos modelos e qual o desempenho dos algoritmos *Mahony* e *Toledo* na detecção de ruído natural. Seção 5 faz uma discussão sobre os experimentos feitos para avaliar o impacto do ruído nos modelos, apresenta os resultados e mostra o *dataset* utilizado para o artigo, além da metodologia usada para obtenção dos resultados obtidos. A seção 5 apresenta uma conclusão sobre os resultados obtidos e propostas para trabalhos futuros sobre o tema.

2. FILTRAGEM COLABORATIVA

Filtragem Colaborativa é um algoritmo de recomendação popular que baseia suas predições e recomendações nas avaliações ou comportamento de outros usuários no sistema. A principal suposição por trás desse método é que as opiniões de outros uários podem ser selecionadas eagregadas de maneira que é possivel ter uma predição razoável das preferências um usuário alvo.

Intuitivamente, assume-se também, que se os usuários concordam sobre a qualidade ou relevância de alguns itens, então eles também vão provavelmente concordar sobre outros itens.

A grande maioria dos algoritmos de CF usados atualmente, operam inicialmente gerando predições de preferência usuário e então produzindo as recomendações deles ranqueando os itens candidatos por preferências previstas.

Os CFs possuem algumas vantagens notáveis:

- Não necessitam de informações sobre conteúdo ou usuários. As *abordagens puras* utilizam apenas as avaliações para fazer a predição;
- Consegue avaliar a experiencia, qualidade e ponto de vista de outras pessoas na predição;
- Consegue sugerir *serendipitous items* através da análise de pessoas do comportamento de usuários semelhantes

Possui também algumas desvantagens, dentre elas:

- Baixa acurácia quando possui poucas informações sobre as avaliações dos usuários(*Cold Start*);
- Ratings são dados explicitamente por usuários, gerando uma base de dados ruidosa por natureza (*ruído natural e/ou ruído malicioso*).

3. PROPOSTA

Em SR é assumido que as avaliações nos *datasets* estão livres de irregularidades. Foi mostrado recentemente que os usuários podem ser inconsistentes quando eles elicitam avaliações para itens, expondo os dados usados nos SR à inconsistências.

Ruído em SR pode ser classificado em 2 categorias principais:

1. *Ruído Malicioso*, associado ao ruído intencionalmente introduzido por um agente externo para influenciar os resultados de um recomendador, e
2. *Ruído Natural*, involutariamente introduzido por usuários, e que também poderia afetar o resultado da recomendação.

O *ruído natural* é inserido sem intenção maliciosa, e, ao contrário do *ruído malicioso*, que já é estudado na literatura há tempos, é um tópico recente.

A identificação do *ruído natural* é mais difícil, pois ele tende a aparecer de diversas formas (ao contrário do malicioso, que costuma estar associado à alguns padrões nos perfis dos usuários).

Neste artigo serão usados dois algoritmos para quantificação do impacto do ruído no modelo, *O'Mahony* e *Toledo*.

3.1 O'Mahony

O algoritmo de *O'Mahony* considera o quão consistente o rating atual para um par usuário-item é em respeito a avaliação prevista que é feita por algum algoritmo de recomendação G.

A *consistência* c de uma avaliação $r_{u,v}$ como o Erro Médio Absoluto (MAE) entre a avaliação atual e a prevista:

$$c(G,T)_{u,v} = \frac{|u, v - p_{u,v}|}{r_{max} - r_{min}}$$

onde $p_{u,v}$ é a avaliação prevista para o par usuário-item (u, v) e r_{min}/r_{max} são o mínimo e máximo dos ratings permitidos, respecivamente. Uma avaliação $r_{u,v}$ é considerada como ruído e excluída do processo de recomendação se:

$$c(G,T)_{u,v} > th$$

onde th é um valor *threshold*.

3.2 Toledo

O algoritmo proposto por *Toledo* inicialmente classifica avaliações, usuários e itens, e que cada item também sua própria tendência de receber avaliações. São identificadas e classificadas conforme a figura 1.

User classes	
Critical user	$\|W_u\| \geqslant \|A_u\| + \|S_u\|$
Average user	$\|A_u\| \geqslant \|W_u\| + \|S_u\|$
Benevolent user	$\|S_u\| \geqslant \|W_u\| + \|A_u\|$
Variable user	Does not satisfy the other user conditions
Item classes	
Weakly-preferred item	$\|W_i\| \geqslant \|A_i\| + \|S_i\|$
Averagely-preferred item	$\|A_i\| \geqslant \|W_i\| + \|S_i\|$
Strongly-preferred item	$\|S_i\| \geqslant \|W_i\| + \|A_i\|$
Variably-preferred item	Does not satisfy the other item conditions

Figure 1: Toledo - Classes de Itens e Usuários

O processo para detecção do ruído assume que para um rating $r(u, i)$ de um dataset, se as classes associadas ao usuário u e o item i pertencem ao meso grupo, então a avaliação deve pertencer a classe de avaliação no mesmo grupo. Se a avaliação não estiver de acordo com a condição, poderia ser uma avaliação ruidosa, e sua transformação poderia mitigar o ruído natural do dataset. Conforme *figura 2*.

Após detectar as possíveis avaliações ruidosas a próxima fase lida com essas avaliações, corrigindo a anomalia ao invés de removê-la para evitar perda de informação. Conforme *figura 3*.

Conforme literatura, não há disponível nenhum *dataset* que não possua ruído natural, logo para testar a acurácia dos algoritmos não é possível usar um *dataset* totalmente apropriado. Para fins desse artigo, iremos considerar que o *dataset* que utilizamos não possui ruído, e a partir do mesmo

Algorithm 1. Detection of possibly noisy ratings

Input: $\mathbf{r} = \{r(u,i)\}$ – set of available ratings, κ_u, ν_u, κ_i, ν_i, κ, ν, – classification thresholds
Output: possible_noise = $\{r(u,i)\}$ – set of possible noisy ratings
01 $W_u = \{\}$, $W_i = \{\}$, $A_u = \{\}$, $A_i = \{\}$, $S_u = \{\}$, $S_i = \{\}$
02 possible_noise = {}
03 for each rating $r(u,i)$
04 if $r(u,i) < \kappa_u$
05 Add $r(u,i)$ to the set W_u
06 else if $r(u,i) >= \kappa_u$ and $r(u,i) < \nu_u$
07 Add $r(u,i)$ to the set A_u
08 else
09 Add $r(u,i)$ to the set S_u
10 if $r(u,i) < \kappa_i$
11 Add $r(u,i)$ to the set W_i
12 else if $r(u,i) >= \kappa_i$ and $r(u,i) < \nu_i$
13 Add $r(u,i)$ to the set A_i
14 else
15 Add $r(u,i)$ to the set S_i
16 end for
17 for each user u and item i
18 Classify it using the associated sets, according to the definitions in Table 2
19 end for
20 for each rating $r(u,i)$
21 if u is critical, i is weakly-preferred, and $r(u,i) >= \kappa$
22 Add $r(u,i)$ to the set possible_noise
23 if u is average, i is averagely-preferred, and ($r(u,i) < \kappa$ or $r(u,i) >= \nu$)
24 Add $r(u,i)$ to the set possible_noise
25 if u is benevolent, i is strongly-preferred, and $r(u,i) < \nu$
26 Add $r(u,i)$ to the set possible_noise
27 end for

Figure 2: Toledo - Detecção de ruído

Algorithm 2. Noisy ratings correction

Input: $\mathbf{r} = \{r(u,i)\}$ – set of available ratings, δ – difference threshold
Output: $\mathbf{r}^* = \{r(u,i)\}$ – set of available ratings corrected
01 poss_noise = possible_noise_detection()
02 for each rating $r(u,i)$ in poss_noise
03 Predict a new rating $n(u,i)$ for user u and item i, using user–user memory-based collaborative filtering with Pearson's correlation coefficient, and using r as the training set
04 if (abs($n(u,i) - r(u,i)$) > δ)
05 Replace $r(u,i)$ by $n(u,i)$ in the original rating set r
06 end for

Figure 3: Toledo - Correção de ruído

iremos inserir ruído malicioso na base e em seguida testar a acurácia dos mesmos usando o *Precision* e *Recall*.

4. EXPERIMENTOS

4.1 Dataset

Para este artigo, foi utilizado o dataset fornecido pelo *Movie Lens Research Project*. *MovieLens* é um SR *web-based* de filmes que começou a operar em 1997.

Consiste de 943 usuários, 1682 filmes e contém 100,000 avaliações no total. As avaliações são baseadas numa escala de 1 a 5.

4.2 Metodologia

A inserção do ruído será feita aleatoriamente iterando de 1 a 10% da base e executando os algoritmos de *Toledo* e *O'Mahony* para detectar o número de avaliações que foi marcada como ruído no mesmo. Essa inserção será feita invertendo-se as notas na base, transformando ratings de acordo com a tabela abaixo:

Rating	Novo Valor
1	5
2	4
3	1
4	2
5	1

Table 1: Troca de Ratings na Inserção de Ruído

4.3 Resultados

5. CONCLUSÃO E TRABALHOS FUTUROS