Comparação entre técnicas de detecção e correção de ruído

Cassiano H. da Silva[1], **Geovani S. Celebrim**[1]

[1]Departamento de Ciência da Computação
Universidade Federal Rural do Rio de Janeiro (UFRRJ)
26.020-740 – Nova Iguaçu – RJ – Brasil

{cassianohsilva, geovanicelebrim}@ufrrj.br

Resumo. *Sistemas de Recomendação têm sido cada vez mais utilizados para promover uma personalização da experiência dos usuários em diversos ambientes. Uma variedade de técnicas são utilizadas para a gerar tais recomendações, entre elas as baseadas em memória e baseadas em conteúdo. Contudo, as avaliações dos usuários podem ser comprometidas por inconsistências no comportamento desses usuários durante o processo de elicitação dessas avaliações. Este trabalho, pois, propõe-se a explorar duas metodologias para detecção e correção dessas inconsistências, também chamadas de ruídos naturais ou maliciosos. A primeira técnica é o algoritmo de Mahony, que detecta o ruído de acordo com um cálculo de consistência de avaliações. A segunda utiliza um método de classificação de notas baseado nos perfis dos usuários e dos itens. Experimentos foram realizados sobre uma base de dados do site MovieLens, contendo 100 mil avalições, comparando o desempenho dessas duas técnicas com algoritmo KNN (baseado em memória) e RSVD (baseado em modelo). Os resultados mostraram que ambos os algoritmos não apresentaram bons resultados na detecção de ruído, não contribuindo para uma melhora significativa na acurácia da previsão após o tratamento dos ruídos detectados.*

1. Definição

Sistemas de Recomendação exploram características e relações entre os usuários e itens com a finalidade de entender como se dá essa relação, para que seja capaz de realizar a recomendação de um novo item para um usuário [Tail 2006]. As informações sobre essas características e relações podem ser capturadas do usuário e do item de forma implícita ou explícita, cada uma com suas vantagens e desvantagens.

Existem diferentes tipos de Sistemas de Recomendação e o que caracteriza essa diferença é a forma como é criada a relação da interação entre os usuários e os itens, bem como a forma que a recomendação é processada. Segundo [Burke 1999], existem basicamente quatro tipos de recomendação: Filtragem Colaborativa, Baseada em Conteúdo, Baseados em Conhecimento e Híbridos. Neste trabalho será explorada a técnica de Filtragem Colaborativa com predição baseada em usuário, introduzida por [Resnick et al. 1994], que recomenda itens baseados na utilização desses pela comunidade de usuários.

Os algoritmos de Filtragem Colaborativa podem ser divididos em dois grandes grupos de técnicas: baseados em memória e baseados em modelo. Os algoritmos baseados em memória, também conhecidos como Baseados em Vizinhança, possuem grande destaque pelo pioneirismo [Goldberg et al. 1992]. Para estes sistemas, o interesse do usuário u no item i é calculado usando as avaliações dos seus usuários mais próximos. Para isso, é

necessário armazenar as avaliações de cada usuário e obter os K usuários mais próximos de u. Calcular essa proximidade de forma eficiente é fundamental para que se obtenha bons resultados.

Outro fator que pode afetar o desempenho de algoritmos de recomendação é a quantidade de ruído presente na base de dados. O ruído ocorre quando uma avaliação não reflete a preferência de um usuário. Esses ruídos podem ser adicionados de maneira natural ou maliciosa.

Ruídos maliciosos são inseridos intencionalmente a fim de tendenciar o algoritmo de recomendação. O algoritmo pode ser tendenciado a recomendar um conjunto de itens para um conjunto de usuário que não possuem grande interesse por esse item.

O ruído natural é inserido devido a variações de critérios de avaliação dos usuários, que podem ser, por exemplo, causado pelo fenômeno chamado de *concept drift* [Schlimmer and Granger 1986, Widmer 1997], onde as preferências dos usuários mudam de acordo com o tempo, levando a avaliações diferentes para itens semelhantes dependendo da época em que foram avaliados.

Neste trabalho serão avaliadas duas técnicas de correção de ruído. A primeira técnica, criada por [O'Mahony et al. 2006], corrige todas as avaliações consideradas como ruidosas de acordo com a avaliação prevista utilizando toda a base de dados. Em [Toledo et al. 2015] são corrigidas somente as avaliações classificadas como possíveis ruídos de acordo com uma heurística que utiliza limiares para classificar usuários, itens e avaliações. Na seção seguinte serão detalhados esses dois algoritmos, apresentando suas características e estratégias.

2. Técnicas de detecção e correção de ruído natural

Em [O'Mahony et al. 2006] é apresentado um estudo sobre detecção de ruído natural e também é proposto um algoritmo para realizar a correção dos ruídos encontrados. No trabalho, a base é dividida entre treino e teste, onde treino é a parte da base utilizada para treinar o modelo de predição e teste é a parte utilizada para testar a qualidade das predições. Considera-se que o teste é genuíno, isto é, que essa parte da base não contém ruído. Partindo desta afirmação, para cada avaliação da base de treino é realizada uma predição e caso a diferença entre o valor predito e o valor real seja maior que um limiar δ o valor da base é substituído pelo valor predito. Este algoritmo é nomeado como *Mahony* neste trabalho.

A técnica apresentada em [Toledo et al. 2015] é bastante semelhante, mas possui uma diferença na avaliação de quais avaliações são ruidosas ou não. Antes de realizar as predições é realizada uma classificação de usuários, itens e avaliações. Usuários e itens podem ser classificados em 4 etiquetas cada:

Etiquetas para os usuários:

1. Benevolente: tendem a avaliar todos os itens positivamente.
2. Mediano: tendem a avaliar os itens com avaliações medianas.
3. Crítico: tendem a avaliar itens negativamente.
4. Hesitante: não pertencem à nenhuma das etiquetas anteriores.

Etiquetas para os itens:

1. Fortemente Preferido: tendem a ser avaliados positivamente.
2. Medianamente Preferido: tendem a ser receber avaliações medianas.
3. Fracamente Preferido: tendem a ser avaliados negativamente.
4. Variavelmente Preferido: não pertencem à nenhuma das etiquetas anteriores.

Os itens são avaliados baseados em dois limiares k e v, onde k separa as classes baixo e médio e v separa as classes médio e alto, onde $k < v$. Esses parâmetros podem ser baseados em valores globais ou na média de avaliações do usuário/item. Com isso, dados um usuário u e um item i, a avaliação $r(u, i)$ pode ser classificada em:

Classes para as avaliações:

1. Baixa: se $r(u, i) < k$;
2. Mediana: se $k \leq r(u, i) < v$;
3. Alta: se $r(u, i) \geq v$.

Para a classificação dos usuários e itens são utilizados os conjuntos de avaliações baixas (W), medianas (A) e altas (S) de acordo com os critérios mostrados na Tabela 1.

Etiqueta do usuário	Etiqueta do item	Condição
Crítico	Fracamente Preferido	$\|W\| \geq \|A\| + \|S\|$
Mediano	Medianamente Preferido	$\|A\| \geq \|W\| + \|S\|$
Benevolente	Fortemente Preferido	$\|S\| \geq \|A\| + \|W\|$
Variável	Variavelmente Preferido	Não se encaixa nas condições anteriores

Tabela 1. Critérios para atribuição de etiquetas aos usuários e itens.

Com essas classificações, são definidas três classes homólogas que associam usuários, itens e avaliações como mostra a Tabela 2. Caso um usuário u e um item i pertençam a classes homólogas, se a avaliação $r(u, i)$ não pertence à uma classe também homóloga, então esta avaliação é considerada como possível ruído. As correções são realizadas com base somente no conjunto de avaliações classificadas como possíveis ruídos. A seleção de quais avaliações devem ser corrigidas diminui o número de avaliações modificadas na base de treino. Este algoritmo é nomeado como *Toledo* neste trabalho.

Grupo	Etiqueta do usuário	Etiqueta do item	Etiqueta da avaliação
Grupo 1	Crítico	Fracamente Preferido	Baixa
Grupo 2	Mediano	Medianamente Preferido	Mediana
Grupo 3	Benevolente	Fortemente Preferido	Alta

Tabela 2. Etiquetas homólogas de usuários, itens e avaliações.

3. Proposta e Metodologia

A proposta deste trabalho é avaliar qualidade dos dois algoritmos apresentados na seção anterior para a base de dados do *MovieLens*[1]. Para realizar as predições, foram utilizados o algoritmo baseado em modelo RSVD[Funk 2006] e o baseado em vizinhança KNN.

[1]Disponível em `http://grouplens.org/datasets/movielens`

Todos os testes foram realizados utilizando a técnica de *cross validation*, onde a base é dividida em n partes iguais e são realizados n testes onde em cada teste uma parte é escolhida para teste e as outras para treino de modo que não haja interseção no conjunto de testes dadas quaisquer execuções distintas. A base de treino é utilizada para realizar as predições e a base de teste é utilizada somente para avaliar a qualidade das predições. Esta técnica garante que toda a base será utilizada para fazer predições, eliminando o possível problema de se escolher uma fatia muito boa ou muito ruim da base. Ao final dos testes é calculada uma média dos resultados de todas execuções.

As perturbações para uma avaliação $r(u, i)$ de um usuário u e item i geram um novo valor aleatório $r'(u, i)$ que é substituído na base de treino de acordo com as seguintes regras:

- Se $r(u, i) < 3$ então $3 < r'(u, i) \leq 5$;
- Se $r(u, i) > 3$ então $1 \geq r'(u, i) < 3$;
- Se $r(u, i) = 3$ então $r'(u, i) \neq 3$.

Para avaliação da qualidade das predições foram utilizadas as métricas *Mean Absolute Error* (MAE), dada pela equação 1 e *Root Mean Squared Error* (RMSE), dada pela equação 2.

$$MAE(f) = \frac{\sum\limits_{r_{ui} \in R_{test}} |f(u, i) - r_{ui}|}{|R_{test}|} \tag{1}$$

$$RMSE(f) = \sqrt{\frac{\sum\limits_{r_{ui} \in R_{test}} (|f(u, i) - r_{ui}|)^2}{|R_{test}|}} \tag{2}$$

Para os testes realizados com o algoritmo KNN, foram utilizados $k = 60$ vizinhos e a função de similaridade Pearson [Resnick et al. 1994], dada pela fórmula 3.

$$w_{uv} = \frac{\sum\limits_{i \in T} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum\limits_{i \in T} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum\limits_{i \in T} (r_{vi} - \bar{r}_v)^2}} \tag{3}$$

Para os testes realizados com o algoritmo RSVD os parâmetros utilizados foram taxa de regularização $\lambda = 0.11$, dimensão de características utilizadas$k = 10$ e taxa de aprendizagem $lrate = 0.005$. Esses parâmetros foram escolhidos com base em testes realizados anteriormente com variação desses parâmetros. Como critério de parada foram utilizados o número de iterações $iteration \leq 250$ e a taxa de melhora de $\Delta > 0.0001$.

4. Experimentos e Resultados

Nos experimentos a base foi dividida em 5 partes, resultando em 80% da base separada para treino e 20% para teste para cada rodada do *cross validation*. Os valores apresentados são uma média dos resultados das 5 execuções.

4.1. Teste de detecção de ruído gerado para o KNN

Neste teste foram gerados ruídos na base de dados e foi realizada uma avaliação dos valores de *precision* e *recall* para o algoritmo de predição. Foi realizada uma perturbação na base de treino. Antes de realizar a perturbação, a base de dados foi previamente corrigida, para posteriormente ser aplicada ao algoritmo de Toledo para a detecção do ruído gerado. A porcentagem de ruído varia de 5% a 25% da base de treino. Os resultados são mostrados no gráfico 1.

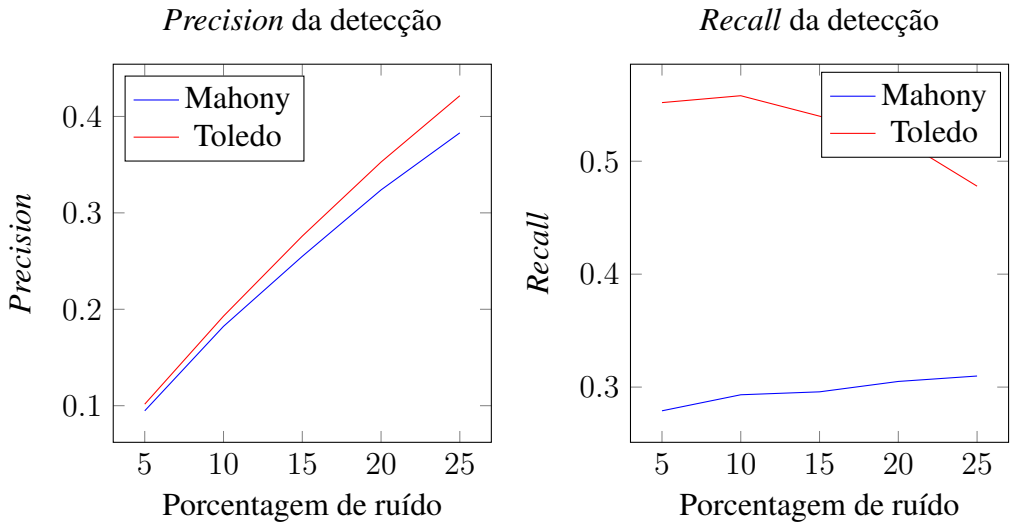

Figura 1. Qualidade das predições.

4.2. Teste de detecção de ruído gerado pelo Mahony variando o threshold utilizando RSVD

Neste teste, inicialmente a base foi corrigida, eliminando, teoricamente, o ruído natural. Após esse procedimento, foi gerado ruído na base de dados e foi realizada uma avaliação dos valores de *precision* e *recall* para os algoritmos. O algoritmo de predição utilizado foi o KNN com 60 vizinhos e a similaridade de Pearson. Foi realizada uma perturbação na base de treino. Antes de perturbar a base, as avaliações foram previamente corrigidas. Após a perturbação, o algoritmo de detecção utilizado foi o Toledo. A porcentagem de ruído varia de 5% a 25% da base de treino. Os resultados são mostrados na Tabela 3.

Limiar	Verdadeiro Positivo	Falso Positivo	Verdadeiro Negativo	Falso Negativo
1	12233	0	64000	3767
2	5874	0	64000	10126
3	1250	0	64000	14750
4	36	0	64000	15964

Tabela 3. Tabela de confusão para detecção de ruído gerado.

4.3. Teste de correções sucessivas utilizando o Toledo

O último teste realizado utilizou o algoritmo Toledo para testar a hipótese de que a correção das avaliações da base de treino pode gerar novos ruídos. Para isso, o algoritmo foi executado 5 vezes e a cada iteração foram avaliadas a quantidade de avaliações

corrigidas. Vale ressaltar que somente 4 das execuções são válidas, dado que na primeira execução não há detecções de ruídos anteriores para realizar comparações. Se uma avaliação foi detectada como ruído em uma iteração $i + 1$ e na iteração i ela não fora detectada como tal, significa que novos ruídos estão aparecendo depois que a base é corrigida. O gráfico 2 mostra a relação entre o número de avaliações detectadas em cada iteração e quantidade de ruídos novos inseridos na matriz de treino.

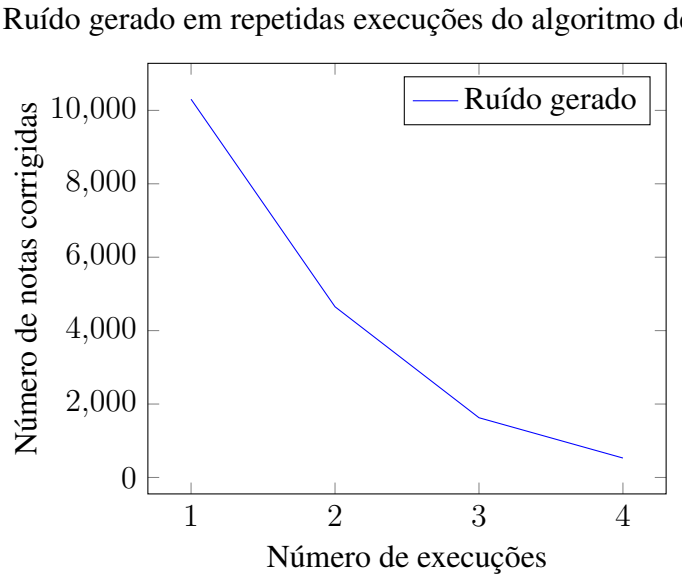

Figura 2. Média da quantidade de ruído gerado na base de treino.

No gráfico 3 são detalhados os valores do MAE e do RMSE após sucessivas execuções do algoritmo de correção.

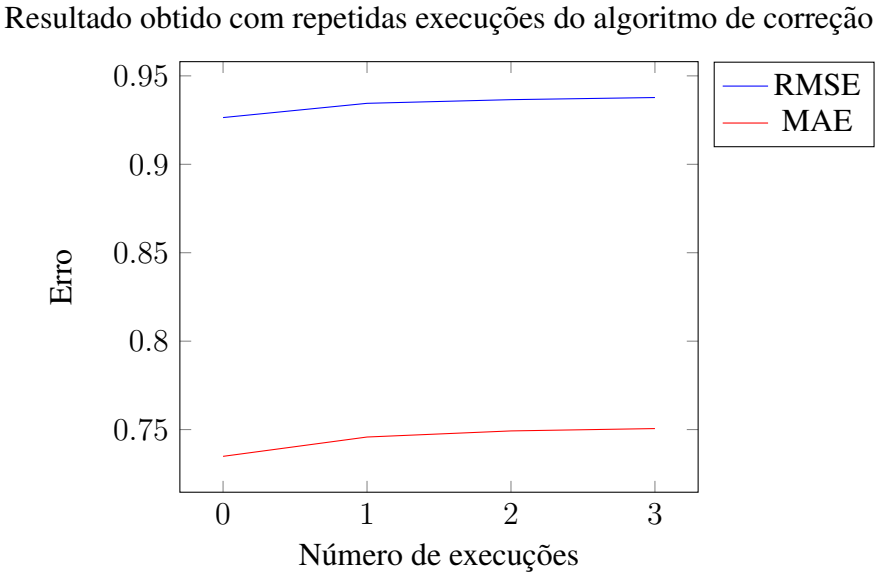

Figura 3. Média da quantidade de ruído gerado na base de treino.

Observa-se que mesmo após as correções, novos ruídos são gerados. Eles vão diminuindo proporcionalmente à quantidade de repetições executadas. Além disso, através do gráfico 3, é possível notar que executar o algoritmo corrigindo a base sucessivas vezes apresentou uma piora nos resultados obtidos com o RSVD.

5. Conclusão

O objetivo deste trabalho foi realizar uma avaliação de técnicas de detecção e correção de ruído na base de dados do *MovieLens*. Foram apresentadas duas técnicas de detecção e correção de ruído que foram analisadas em diferentes aspectos.

Esses resultados mostram que informações avaliações discrepantes podem afetar negativamente o desempenho dos Algoritmos de Recomendação. Dependendo de qual o domínio do problema, pode ser necessária a adição de algoritmos de detecção e correção de ruído para se alcançar boas predições.

Os algoritmos avaliados não apresentaram bons resultados em grande parte dos casos e pôde-se observar que a execuções consecutivas desses pode acarretar na inserção de mais ruído na base de dados.

Como trabalhos futuros pretende-se utilizar heurísticas para melhorar a avaliação de limiares para classificação dos usuários, itens e avaliações no algoritmo Toledo. Uma possível heurística é fazer as classificações baseados na vizinhança do usuário/item e não de modo global para adaptar melhor as preferências dos usuários/itens aos dos itens/usuários.

Referências

Burke, R. (1999). Integrating knowledge-based and collaborative-filtering recommender systems. In *Proceedings of the Workshop on AI and Electronic Commerce*, pages 69–72.

Funk, S. (2006). Netflix update: Try this at home (december 2006).

Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70.

O'Mahony, M. P., Hurley, N. J., and Silvestre, G. (2006). Detecting noise in recommender system databases. In *Proceedings of the 11th international conference on Intelligent user interfaces*, pages 109–115. ACM.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM.

Schlimmer, J. C. and Granger, R. H. (1986). Incremental learning from noisy data. *Machine learning*, 1(3):317–354.

Tail, L. (2006). Why the future of business is selling less of more.

Toledo, R. Y., Mota, Y. C., and Martínez, L. (2015). Correcting noisy ratings in collaborative recommender systems. *Knowledge-Based Systems*, 76:96–108.

Widmer, G. (1997). Tracking context changes through meta-learning. *Machine Learning*, 27(3):259–286.