Comparativo Entre Algoritmos para Sistemas de Recomendação de Filmes baseados em Filtragem Colaborativa*

André Leitzke Junior[1]

[1]Instituto de Informática – Universidade do Vale do Rio dos Sinos (Unisinos)
Avenida Unisinos, 950 – 93.022-000 – São Leopoldo – RS – Brazil

leitzke.andre@gmail.com

Abstract. *This paper describes a comparison between two collaborative filtering recommendation algorithms using Apache Mahout and the MovieLens data set.*

Resumo. *Este artigo descreve um comparativo de algoritmos de recomendação baseados em filtragem colaborativa utilizando a engine Apache Mahout sobre o conjunto de dados do MovieLens.*

1. Sistemas de Recomendação

Os sistemas de recomendação são aplicações que visam, com base em caracterísiticas conhecidas como o perfil estimado do usuário a partir de avaliações, no caso da filtragem colaborativa, ou dados e metadados de itens, no caso de filtragem baseada em conteúdo, auxiliar a busca por itens relevantes para um usuário em um dado momento.

São amplamente utilizados por sites ligados a conteúdo multimídia, como filmes e músicas, assim como de modo geral em sites de e-commerce, devido à inviabilidade de que um usuário busque item a item numa listagem o conteúdo relevante para si.

Em se tratando especificamente de conteúdo multimídia, o mais comum é que seja utilizada a filtragem colaborativa, pois há relativamente pouco conteúdo disponível, em geral na forma de metadados, sobre os itens a serem recomendados, tornando assim a recomendação baseada em conteúdo um tanto ineficaz dependendo de como é utilizada.

2. Colaboração

As principais contribuições deste artigo se dão através:

- da apresentação de dois algoritmos de filtragem colaborativa, ambos provenientes da ferramenta Taste, parte integrante da engine de recomendação Apache Mahout;
- de um breve comparativo da eficácia dos algoritmos apresentados dentro do conjunto de dados utilizado.

3. Domínio da Aplicação

Este artigo apresenta o estudo de recomendação para filmes através do uso da filtragem

* Adaptação para OpenOffice.org 1.1 feita por Roland Teodorowitsch (roland@ulbra.tche.br) em 29 mar. 2005.

colaborativa. De forma simplificada, é ilustrada a implementação de dois algoritmos de filtragem colaborativa, ambos atuando sobre um conjunto de dados que representa avaliações de filmes, o MovieLens Data Set, composto por 100 mil avaliações dadas por aproximadamente 1000 usuários a um conjunto de filmes.

Sendo o foco principal o estudo e a apresentação dos algoritmos de recomendação, não foi investido tempo algum no desenvolvimento de uma interface. Toda a interação com o protótipo é realizada através do prompt de comando.

4. A Engine Apache Mahout

Apache Mahout é uma biblioteca que possui uma compreensiva lista de algoritmos focados em aprendizado de máquina (machine learning), classificação, agrupamento (clustering) e recomendação.

A biblioteca é focada em prover escalabilidade, tanto de funcionalidades, através de sua licença que prevê e é bastante aberta quanto a extensões e trabalhos derivados, quanto com relação aos conjuntos de dados suportados, trabalhando sobre Apache Hadoop para prover escalabilidade através de computação e armazenamento distribuídos.

Diversas ferramentas individuais foram descontinuadas e agregadas ao projeto da engine, como, por exemplo, a engine de filtragem colaborativa Taste.

5. A Implementação do Algoritmo de Recomendação

A aplicação foi desenvolvida em Java (Java JDK 1.6), utilizando a biblioteca Apache Mahout (versão 0.6-SNAPSHOT).

O processo de utilização dos algoritmos segue um padrão bem definido, graças à boa organização da biblioteca. No caso do nosso protótipo, importando-se os dados de recomendação de um arquivo .csv, o processo se dá da seguinte forma:

1. importa-se o arquivo em um DataModel;
2. instancia-se o objeto do recomendador (da classe SlopeOneRecommender ou KnnItemBasedRecommender, dependendo do caso);
3. lê-se o id do usuário para o qual deve ser realizada a recomendação;
4. solicita-se ao objeto recomendador os itens recomendados àquele usuário;
5. itera-se pelas recomendações, exibindo-as.

```
File ratingsFile = new File("/home/andre/datasets/dummy-100k.csv");
DataModel model = new FileDataModel(ratingsFile);
CachingRecommender recommender = new CachingRecommender(new KnnItemBasedRecommender(model,
        new PearsonCorrelationSimilarity(model),
        new ConjugateGradientOptimizer(), 5));

long userId = Prompt.ReadUser();

List<RecommendedItem> recommendations = recommender.recommend(userId, 3);
```

Figura 1 - Exemplo da implementação de recomendação com k-Nearest Neighbor

6. Algoritmos Utilizados

6.1. Slope One

Slope One é considerada uma das mais simples técnicas de recomendação não-triviais, baseadas em conteúdo. É uma técnica de implementação bastante simples, mas que possui precisão competitiva com algoritmos muito mais "caros" (em termos de capacidade computacional).

O cálculo da recomendação mostra como é uma técnica simples, mas, apesar disso, em diversas ocasiões mostrou-se mais eficiente que cálculos mais complexos, que utilizem regressão linear.

Exemplo de cálculo do algoritmo básico:

1. Usuário 1 deu notas 2 para o item A e 4 para o item B.
2. Usuário 2 deu nota 2.5 para o item A.
3. A nota do usuário 2 para o item B seria igual a 4.5 (4 - 2 + 2.5)

O algoritmo real utilizado pelo Apache Mahout apresenta algumas diferenças com relação ao básico, logo os valores não seguem diretamente esta lógica de cálculo, mas são apenas otimizações, na maioria dos casos, a pontuação esperada é próxima à obtida por este cálculo.

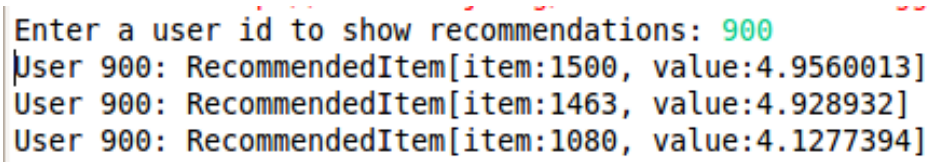

Figura 2 - Recomendação utilizando Slope One

6.2. K-Nearest Neighbor (ou k-Nearest Neighborhood)

Esta é uma das técnicas mais comumente utilizadas na recomendação. Pode-se dizer que o k-NN é uma técnica que classifica os objetos com base nas propriedades dos vizinhos mais próximos, ou seja, através dos perfis mais compatíveis ao do usuário, será definida a nota estimada para um certo item.

O k-NN é considerado um método computacionalmente custoso, dado que não depende de uma fase de preparação, todo o processamento é realizado na hora da geração da recomendação, além disso, depende fortemente de uma grande quantidade de dados para recomendações eficazes.

No caso desta implementação, a similaridade é estimada através do Coeficiente de Correlação de Pearson, na forma de um objeto da classe PearsonCorrelationSimilarity, que é passado para o construtor do objeto recomendador.

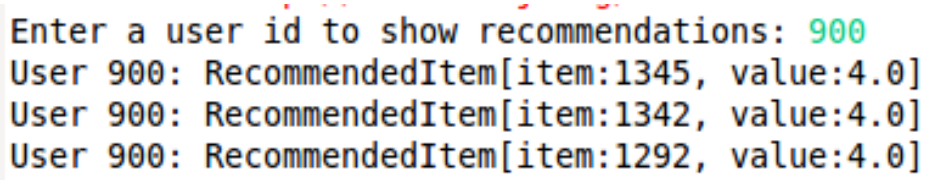

Figura 3 - Recomendação usando k-Nearest Neighborhood

7. Metodologia para a comparação

Após o teste dos algoritmos com o conjunto de 100 mil recomendações, o serão utilizados valores de um conjunto de 80% dos dados, onde serão geradas sem ordem lógica específica algumas recomendações com ambos algoritmos para itens que tenham qualificação existente nos outros 20%, a fim de posteriormente realizar o cálculo da média absoluta de erro, que será comparada entre os dois algoritmos.

Foi selecionado um usuário com 238 avaliações no conjunto de dados de teste, além de 24 no conjunto de dados para comparação, a fim de se ter um número relativamente grande de dados servindo como entrada para a recomendação.

8. Comparativo

Realizado o comparativo, diversos itens não foram calculados através do uso do algoritmo k-NN. Sendo assim, foi calculada a média absoluta de erro com base nos itens cuja estimativa de rating pode ser calculada.

Conforme apontado por diversos autores, o algoritmo Slope One obteve predições mais próximas que o k-NN, apresentando média de erro de 0,83 pontos, contra 1,13 do outro algoritmo.

A tabela com os comparativos encontra-se como um anexo, no final do artigo.

9. Conclusão

Através da comparação destes dois algoritmos, foi possível aferir de forma prática muito do que é conhecido sobre sistemas de recomendação baseados em filtragem colaborativa, como a necessidade de uma grande massa de dados para estimativas mais corretas, além do fato de um algoritmo relativamente simples, como o Slope One, conseguir ser bastante preciso, oferecendo estimativas mais próximas da realidade que o outro algoritmo testado, além da menor necessidade computacional, que em diversos momentos torna-se algo vital para o sucesso de um sistema, ao traduzir-se em respostas mais rápidas para o usuário de um e-commerce, por exemplo.

10. Perspectivas

Apesar de simplificada, esta comparação dificilmente é encontrada, são feitos poucos benchmarks entre os diferentes algoritmos de recomendação. Um próximo passo natural seria expandir o teste destes dois algoritmos, aplicando-os aos datasets de 1 milhão e 10 milhões de recomendações do MovieLens. Também pode-se pensar em expandir para outros algoritmos de recomendação, ou mesmo investir um tempo maior na criação de uma suíte de benchmark extensível, que permita a comparação rápida de diversos algoritmos contra um mesmo dataset qualquer.

11. Referências bibliográficas

- Apache Mahout - http://mahout.apache.org/
- k-Nearest Neighborhood - http://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm
- Slope One Algorithm - http://en.wikipedia.org/wiki/Slope_One

12. Trabalhos relacionados

- Cacheda, F., Carneiro, V., Fernandez, D., and Formoso, V. 2011. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. ACM Trans. Web 5, 1, Article 2 (February 2011), 33 pages - HTTP://ART.CASE.EDU/339.439.F11/INTERESTING%20PAPERS/2011.COLLABORATIVEFILTERING.PDF

13. Anexos

User 437					
Item	Rating real	KNN	Desvio	Slope One	Desvio
14	5.00	NaN		3.75	1.25
86	4.00	NaN		3.66	0.34
118	2.00	NaN		2.93	0.93
165	4.00	NaN		4.06	0.06
173	4.00	NaN		4.10	0.10
190	3.00	NaN		4.13	1.13
197	5.00	3.20	1.80	3.99	1.01
239	4.00	NaN		3.42	0.58
418	3.00	NaN		3.44	0.44
419	5.00	NaN		3.63	1.37
451	5.00	NaN		3.10	1.90
499	5.00	NaN		3.93	1.07
584	3.00	NaN		3.34	0.34
640	1.00	NaN		2.95	1.95
652	4.00	NaN		3.80	0.20
655	4.00	NaN		3.68	0.32
665	2.00	NaN		2.55	0.55
696	3.00	NaN		2.77	0.23
708	4.00	NaN		3.23	0.77
755	3.00	3.20	0.20	3.12	0.12
1,113	4.00	NaN		3.32	0.68
1,121	5.00	NaN		3.77	1.23
1,153	5.00	NaN		3.39	1.61
1,262	3.00	4.40	1.40	3.67	0.67
1,404	2.00	NaN		3.00	1.00
Mean Absolute Error			1.13		0.83

Figura 4 - Resultados das predições e médias de erro