Estudo Comparativo de Bibliotecas para Sistemas de Recomendação em Python

Lorenzo Dalla Corte Danesi[1], **Daniel Lichtnow**[1]

[1]Colégio Politécnico – Universidade Federal de Santa Maria (UFSM)
CEP 97105-900 – Santa Maria – RS – Brasil

lorenzo.danesi@acad.ufsm.br, daniel.lichtnow@ufsm.br

Abstract. *This paper presents an analysis of Python language libraries that implement Collaborative Filtering algorithms used in Recommender Systems. Using two libraries, Surprise and LensKit, the K-Nearest Neighbors (K-NN) and Slope One algorithms are explored and comparative tests are carried out to evaluate libraries.*

Resumo. *Este artigo apresenta uma análise de bibliotecas em linguagem Python que implementam algoritmos de Filtragem Colaborativa usados em Sistemas de Recomendação. Por meio de duas bibliotecas, Surprise e LensKit, são explorados os algoritmos K-Nearest Neighbors (K-NN) e Slope One e realizados testes comparativos para avaliar as bibliotecas.*

1. Introdução

Com a popularização da Internet, ocorreu uma crescente oferta de conteúdo e de produtos para os usuários. O volume de itens oferecidos na Web demandou a criação de ferramentas que facilitassem a identificação de itens de interesse dos usuários. Estas ferramentas são referenciadas como Sistemas de Recomendação e são hoje usadas no mercado e na área acadêmica devido à capacidade desses sistemas em lidar com a abundância de dados e sobrecarga de informações [Adomavicius and Tuzhilin 2005].

Com isso, diversas bibliotecas foram desenvolvidas para implementar algoritmos avançados, proporcionando ferramentas que facilitam a manipulação de dados, realizam operações complexas e aprimoram a qualidade das recomendações. Entre as soluções disponíveis, destacam-se as bibliotecas *Surprise* e *LensKit*, ambas escritas em Python, que oferecem funcionalidades robustas para o desenvolvimento de Sistemas de Recomendação eficientes. Este artigo explora e avalia o uso dessas bibliotecas por meio de testes realizados com *datasets*, analisando sua aplicabilidade, desempenho e contribuições para o aprimoramento das técnicas de recomendação. Um comparativo entre bibliotecas realizado em [Said and Bellogín 2014] utilizava maior número de métricas e *datasets*, porém não avalia o *Surprise* e utiliza a versão Java do *LensKit*.

2. Sistemas de Recomendação

Um Sistema de Recomendação pode ser definido como uma ferramenta computacional que sugere itens úteis para usuários [Ricci et al. 2010]. Para esses sistemas, existem três abordagens clássicas: Filtragem Baseada em Conteúdo, Filtragem Colaborativa e Híbrida. Em função de sua ampla utilização e de possuir vários dos seus algoritmos implementados em diversas bibliotecas, este trabalho foca na abordagem de Filtragem Colaborativa.

Essa abordagem considera as opiniões/avaliações de usuários sobre itens e a geração de recomendações envolve a comparação do grau de concordância entre as avaliações e não depende da representação dos itens como na Baseada em Conteúdo e frequentemente em abordagens Híbridas.

A Filtragem Colaborativa possui duas variações básicas: *user-user* e *item-item*. Na variação *user-user*, a recomendação é gerada a partir da similaridade entre os usuários, sendo a similaridade calculada a partir das avaliações que os usuários deram para itens [Resnick et al. 1994]. Já na variação *item-item* são recomendados itens com base na similaridade entre os próprios itens, sendo considerados itens similares aqueles que tiveram avaliações similares feitas por diferentes usuários [Sarwar et al. 2001].

Na variação *user-user*, inicialmente compara-se um usuário com outros a partir das notas atribuídas aos itens. O cálculo dessa similaridade é feito utilizando um coeficiente de correlação (e.g. cosine, Pearson). Em seguida, selecionam-se os vizinhos mais próximos (usuários com padrões de avaliação semelhantes) e, finalmente, para itens não avaliados por um usuário é estimada a nota com base nas avaliações dos vizinhos selecionados para este item (e.g. média ponderada das avaliações dos vizinhos) e recomendados aqueles com nota estimada maior. Já na variação *item-item* são comparados itens, isto é, as notas dadas a eles, sendo também utilizado um coeficiente de correlação e considerado que se muitos usuários atribuírem notas similares para dois itens, esses são similares.

Outro exemplo é o *Slope One*, um algoritmo que apresenta a proposta de implementar Sistemas de Recomendação de Filtragem Colaborativa *item-item* de forma mais simples e escalável. Basicamente no *Slope One* é calculada a diferença média entre as avaliações de dois itens A e B por todos usuários que avaliaram estes dois itens. Esta diferença é considerada para prever a avaliação de A por um usuário que avaliou apenas o item B [Lemire and Maclachlan 2005].

3. Avaliação de Bibliotecas

Foram identificadas duas bibliotecas como as mais destacadas dentre as que implementam algoritmos referenciados na seção 2 e variações destes:

- **Surprise** Para o desenvolvimento e avaliação de Sistemas de Recomendação, esta biblioteca oferece uma variedade de algoritmos de predição de Filtragem Colaborativa [Hug 2020]. Também possui funções para algoritmos baseados em similaridade, como o K-NN, além de implementações do algoritmo *Slope One* para a realização de predições.
- **LensKit for Python (LKPY)** Lançada em 2010 como uma biblioteca em Java, visava apoiar pesquisas e estudos em Sistemas de Recomendação. Em 2020, teve sua versão lançada em Python, chamada *LensKit for Python* (LKPY), com implementações de algoritmos clássicos de Filtragem Colaborativa, além de implementar funções de avaliação e ferramentas para integração com outros *softwares* da mesma linguagem [Ekstrand 2020].

Para a avaliação das bibliotecas foram utilizados *datasets* disponibilizados pelo MovieLens, uma plataforma não comercial de recomendação de filmes personalizada, com dados coletados em diferentes períodos de tempo [Harper and Konstan 2015]. Esses conjuntos de dados incluem informações sobre usuários, filmes e avaliações com tamanhos que variam de 100 mil a 20 milhões de avaliações com uma escala de 1 a 5 estrelas.

Ainda, foi usado o Google Colab para executar os experimentos, um ambiente que oferece acesso a GPUs potentes e alta capacidade de memória. A Figura 1 apresenta o código usado para rodar um dos algoritmos de recomendação da biblioteca *Surprise*.

```
import pandas as pd
from surprise import Dataset
from surprise import KNNBasic
from surprise.model_selection import cross_validate

# carregar o dataset do movielens-100K
data = Dataset.load_builtin('ml-100k')

sim_options = {
    'name': 'pearson',
    'user_based': True  # baseado em usuários
}
model = KNNBasic(sim_options=sim_options)

# avaliar o modelo usando validação cruzada
results = cross_validate(model, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

Figura 1. Exemplo de uso da biblioteca *Surprise*.

Os valores apresentados na Tabela 1 indicam as diferenças de desempenho de cada algoritmo de recomendação nas bibliotecas com os *datasets*. Para avaliar o desempenho foi medido o tempo, obtido somando os tempos de execução para cada *fold* do conjunto de testes. Além disso, a partir da validação cruzada (5 *folds*), são apresentados os valores de métricas de avaliação utilizadas, taxas de MAE *(Mean Absolute Error)*, ou Erro Médio Absoluto, e RMSE *(Root Mean Square Error)*, ou Raiz do Erro Médio Quadrático, ambas indicam a média dos erros de previsão do modelo, porém, o RMSE penaliza maiores desvios entre as previsões realizadas e as notas efetivamente atribuídas a um filme.

Tabela 1. Resultados do testes dos algoritmos de recomendação e bibliotecas.

	Bibliotecas/algoritmos	**Tempo**	**MAE**	**RMSE**	**Bibliotecas/algoritmos**	**Tempo**	**MAE**	**RMSE**
	Surprise				**LKPY**			
ml-100K	User- User	18 s	0.80	1.01	User-User	2 s	0.75	0.96
	Item- Item	23 s	0.81	1.02	Item-Item	41 s	0.71	0.91
	Slope One	18 s	0.74	0.94	ND	ND	ND	ND
ml-1M	User- User	18 min	0.77	0.97	User-User	45 s	0.71	0.91
	Item- Item	7 min	0.78	0.99	Item-Item	5 min	0.68	0.88
	Slope One	6 min	0.71	0.90	ND	ND	ND	ND
ml-10M	User- User	ND	ND	ND	User-User	53 min	0.67	0.87
	Item- Item	80 min	0.69	0.88	Item-Item	70 min	0.65	0.85
	Slope One	88 min	0.66	0.86	ND	ND	ND	ND
ml-20M	User- User	ND	ND	ND	User-User	212 min	0.66	0.87
	Item- Item	180 min	0.67	0.87	Item-Item	213 min	0.64	0.84
	Slope One	139 min	0.65	0.85	ND	ND	ND	ND

O uso da biblioteca *Surprise*, exemplificado na Figura 1, apresentou maior facilidade em relação à LKPY, devido à sua documentação, integração com *datasets* e disponibilidade da função de validação cruzada. No entanto, o desempenho dos algoritmos, especificamente *user-user*, apresentou maior custo computacional ocasionando problemas no uso de memória, não sendo possível obter os resultados com *datasets* de 10 e 20 milhões de avaliações. Cabe salientar que a biblioteca *Surprise* possui implementação do algoritmo *Slope One*, ausente na biblioteca LKPY.

Sobre as variações de desempenho entre as duas bibliotecas, nos casos em que foi possível rodar o mesmo algoritmo com o mesmo tamanho de *dataset* em cada uma das bibliotecas, foi avaliado se as diferenças eram estatisticamente significativas considerando os resultados na Tabela 1. Foi usado o Teste t Independente (MAE e RMSE - distribuição normal) e o teste de Mann-Whitney (Tempo). Os resultados são apresentados na Tabela 2, sendo que em todos os p-valor não foi inferior a 0.05, indicando que não há diferença significativa. Porém pode-se verificar que há uma vantagem da LKPY quanto às métricas MAE e RMSE que pode ser considerada marginalmente significativa.

Tabela 2. Avaliação estatística de performance.

Variável	Tempo	MAE	RMSE
p-valor	0.8182	0.0567	0.0723

4. Considerações Finais

As bibliotecas demonstraram ser úteis para a área de Sistemas de Recomendação, sendo a *Surprise* uma biblioteca que apresenta, no momento, maior facilidade de uso. A análise da performance (tempo, MAE e RMSE) deve ser melhor avaliada em trabalhos futuros.

Referências

Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749.

Ekstrand, M. D. (2020). Lenskit for python: Next-generation software for recommender systems experiments. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*.

Harper, F. M. and Konstan, J. A. (2015). The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19.

Hug, N. (2020). Surprise: A python library for recommender systems. *Journal of Open Source Software*, 5(52):2174.

Lemire, D. and Maclachlan, A. (2005). Slope one predictors for online rating-based collaborative filtering. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pages 471–475. SIAM.

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.

Ricci, F., Rokach, L., and Shapira, B. (2010). Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer.

Said, A. and Bellogín, A. (2014). Comparative recommender system evaluation: benchmarking recommendation frameworks. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 129–136.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295.