Análise de Sistemas de Recomendação para o Catálogo de Jogos Steam

LUCAS CIZIKS* and PEDRO MAÇONETTO*, Instituto de Ciências Matemáticas e de Computação, Brasil

Fig. 1. Catálogo de jogos da Steam, 2022.

Desenvolvimento, avaliação e comparação de diferentes abordagens de Sistemas de Recomendação para sugestão e ranqueamento de jogos com base no catálogo da plataforma de compra e venda de jogos online Steam.

CCS Concepts: • **Information systems** → *Recommender systems.*

Additional Key Words and Phrases: recommender systems, datasets, data science, machine learning

ACM Reference Format:
Lucas Ciziks and Pedro Maçonetto. 2022. Análise de Sistemas de Recomendação para o Catálogo de Jogos Steam. 1, 1 (December 2022), 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUÇÃO

Um **Sistema de Recomendação** é responsável por calcular e sugerir conteúdos relevantes ao usuário, com base nos metadados dos itens, interações históricas do usuário com outros itens ou até mesmo algum conhecimento do domínio de interesse [3]. Desse modo, não é ao acaso que empresas de *e-commerce* e serviços de *streaming* investem intensamente em aprimorar as recomendações por trás de seus produtos (tome como exemplo o Netflix Prize), procurando entender as melhores abordagens e combinações em seu cenário. Isso porque melhores recomendações levam a um número maior de vendas e/ou adêrencia aos serviços, o que, consequentemente, traz um maior retorno financeiro e lucro às empresas [8].

*Ambos autores contribuiram igualmente para o trabalho.

Authors' address: Lucas Ciziks, luciziks@usp.br; Pedro Maçonetto, pedromaconetto@usp.br, Instituto de Ciências Matemáticas e de Computação, Av. Trab. São Carlense, 400, São Carlos, São Paulo, Brasil, 13566-590.

Manuscript submitted to ACM

Nesse cenário, é possível encontrar diversas abordagens para um mesmo problema, e, por se tratar de uma área ainda muito recente, novos métodos e modelos são desenvolvidos frequentemente. Algumas abordagens mais sofisticadas realizam o ranqueamento dos itens por meio de um modelo treinado através de aprendizado supervisionado [5]. Nesse contexto, é interessante analisar e comparar o desempenho de diferentes famílias de algoritmos de treinamento *Learning to Rank* (**LTR**), verificando qual deles se adequa melhor ao problema estudado.

Esses algoritmos podem ser distinguidos pela maneira que lidam com a lista de recomendação durante seu treinamento [3]. Os algoritmos **Pointwise** são caracterizados pelo cálculo de uma nota (*score*) para todos os itens da base, ordenando-os na lista de recomendação final. Enquanto isso, o **Pairwise** compara e ordena os itens de forma pareada, otimizando a geração lista de recomendação através de um número mínimo de inversões. Já os algoritmos de **Listwise** analisam a lista como uma única unidade, otimizando e ranqueando de acordo com a relevância e posição dos itens [3].

2 METODOLOGIA

2.1 Objetivo

O presente projeto possui como principal objetivo estudar e analisar diferentes algoritmos e abordagens para ranqueamento em um contexto ainda pouco explorado de **compra e venda de jogos** da plataforma Steam. Essa análise será feita com base na comparação das 3 diferentes famílias de algoritmos **LTR**. Para isso, utilizaremos bases de dados do Kaggle, que reúnem informações relevantes sobre os jogos comercializados e várias avaliações de usuários dentro da plataforma.

Portanto, o problema apresentado pode ser decomposto em:

- Amostrar e tratar as bases de dados em investigação;
- Estabelecer e utilizar um algoritmo representante de cada família **LTR** que se encaixe ao problema estudado;
- Comparar a performance e eficácia de cada sistema escolhido;

Assim, ao final do projeto, teremos diferentes métodos para recomendação de jogos testados, com suas respectivas métricas calculadas e comparadas.

2.2 Especificação

Para implementar e testar os recomendadores, optamos por utilizar a linguagem **Python**, já que, por se tratar de um problema com necessidade de manipulação massiva de dados, a linguagem simplifica tais operações, utilizando ferramentas como Pandas e NumPy. Além disso, há a biblioteca CaseRecommender que facilita o uso dos principais algoritmos de recomendação[2].

As bases de dados a serem utilizadas para treinar e avaliar os modelos são:

- Steam Reviews 2021: Avaliações reais de jogos na plataforma Steam em 2021;
- Steam Store Games: Informações e metadados sobre os jogos disponíveis na plataforma Steam.

Para usufruí-las, será necessário realizar um processo de amostragem e limpeza, devido ao tamanho das bases reais (8 GB).

2.3 Implementação

Para analisar e estudar cada um dos tipos de algoritmos **LTR**, testaremos os seguintes algoritmos:

- **Pointwise**: Filtragem Colaborativa Baseada em Modelo (SVD Otimizado [4]) e Filtragem Baseada em Conteúdo (Item Attribute KNN);
- **Pairwise**: Bayesian Personalized Ranking (BPR) [6];
- **Listwise**: ListNet [1].

Para comparar o desempenho desses diferentes métodos de recomendação, aplicaremos o método offline de avaliação, dividindo uma parcela das avaliações para a realização dos testes e a outra para treinamento e aprimoramento do modelo (**Hold-out**). A **Acurácia** será avaliada por meio do **ranqueamento dos jogos**, medido por meio de métricas disseminadas como a precisão, AP e MAP do sistema [7]. Além disso, avaliaremos a diversidade do ranqueamento gerado, medido pela métrica Intra-List Similarity (ILS) [9].

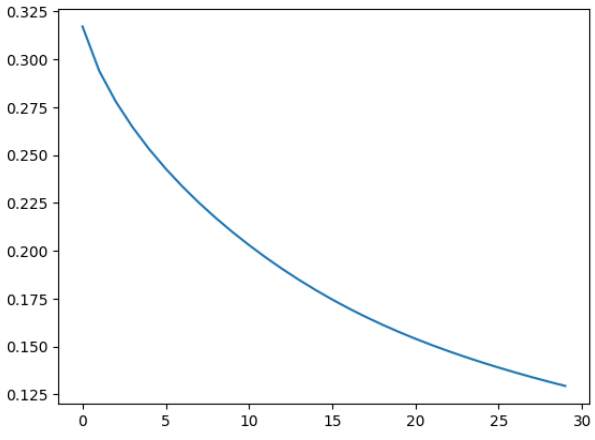
3 DESENVOLVIMENTO

Como descrito na seção de Metodologia, a implementação do projeto foi realizada em um Notebook (.ipynb) da linguagem Python. A amostragem, limpeza dos dados, aplicação dos recomendadores e avaliação dos resultados está disponível no repositório do Github Recommender System Catalogue. Uma explicação da estruturação do desenvolvimento do projeto está disponível em https://drive.google.com/file/d/1bgRsMiRHAuxVPhUMXUYka_G6voB1aIoS/view?usp=sharing.

3.1 Pointwise

Para testar a eficiência da Filtragem Colaborativa Baseada em Modelo, utilizamos o algoritmo **SVD Otimizado**. Esse algoritmo reúne as vantagens dos métodos de fatoração de matriz, os quais encontram padrões de interação latentes no conjunto de dados, mas também resolve os problemas de lentidão do SVD através do Gradiente Descendente Estocástico, que otimiza a decomposição pela minimização do RMSE [4]. Em nossa implementação, utilizamos 5 fatores e 30 interações para treinar o modelo. Com esses parâmetros, a média do erro erro quadrático médio RMSE de cada usuário foi de 0,215. O MAP@10 foi de 0,0621, enquanto o erro do modelo convergiu como é possível observar no gráfico abaixo:

Fig. 2. Training error



Para a Filtragem Baseada em Conteúdo, utilizamos o algoritmo já implementado pela biblioteca CaseRecommender **ItemAttributeKNN**. Primeiramente, testamos seu uso com o cálculo de similaridade interno da própria biblioteca, atingindo um MAP@10 de 0,0371. Posteriormente, calculamos a similaridade através da medida Cossine Similarity [3], utilizando apenas as informações de **Categoria** e **Genêro** do jogo. Com isso, obtivemos um MAP inferior de 0,030120.

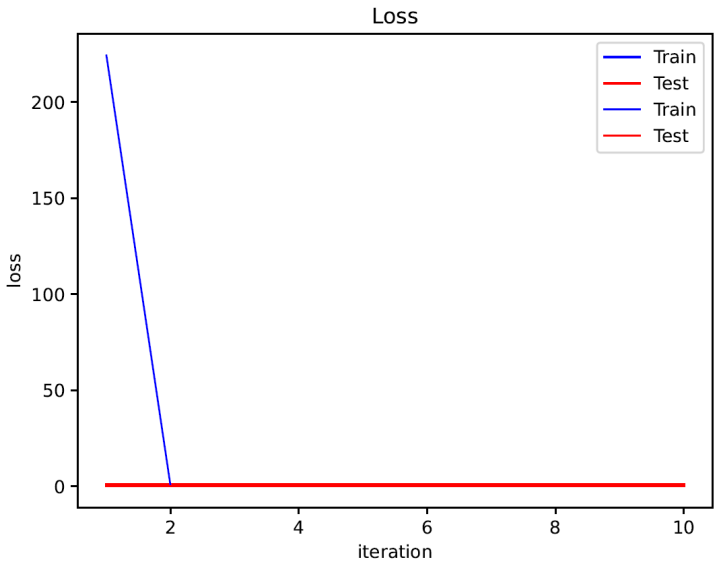
3.2 Pairwise

Como exemplo de um algoritmo Pairwise, utilizamos o **Bayesian Personalized Ranking** ou simplesmente **BPR**. O BPR é normalmente utilizado em dados de feedback implícito, o que pode ser uma vantagem para a base de dados em investigação, uma vez que as avaliações são binárias, recomendado ou não. Aplicando o próprio algoritmo do CaseRecommender, que une o BPR e a fatoração de matriz, verificou-se o desempenho para diferentes números de fatores: 5, 15 e 30. Os resultados obtidos serão apresentados na seção de Avaliação.

3.3 Listwise

Para testar um algoritmo da família Listwise, utilizamos o ListNet [1]. Devido à complexidade desses algoritmos e à dificuldade em encontrar implementações, pois muitas delas estavam defasadas, optamos por implementar um algoritmo similar ao encontrado nesse repositório focado em Learning to Rank e baseado no artigo original. Entretanto, não conseguimos avaliar outra métrica além da MAP para esse algoritmo. Abaixo está o gráfico de "loss"ou perda do modelo. Como é possível observar, o algoritmo convergiu rapidamente, antes das 10 interações:

Fig. 3. Listwise loss



4 AVALIAÇÃO

A fim de avaliar cada variação dos algoritmos LTR utilizados, as métricas de **MAP** para 1, 3, 5 e 10 jogos no topo do ranqueamento foram calculadas e armazenadas. Em paralelo, também avaliamos a diversidade de cada ranqueamento através da métrica **ILS**. Os resultados obtidos estão sumarizados na seguinte tabela:

	Pointwise			Pairwise			Listwise
	SVD Otimizado	IAKNN CaseRecommender	IAKNN Cossine Similarity	BPRMF 5 Fatores	BPRMF 15 Fatores	BPRMF 30 Fatores	ListNet
MAP@1	0.065371	0.029418	0.019842	0.168172	0.166845	0.168071	0.825764
MAP@3	0.062340	0.029394	0.019842	0.163876	0.163621	0.164852	0.825764
MAP@5	0.062200	0.029394	0.019842	0.163872	0.163588	0.164856	0.825764
MAP@10	0.062188	0.029394	0.019842	0.163868	0.163588	0.164856	0.825764
ILS	10.705737	17.714538	15.260545	8.090326	8.048933	7.898754	Não se aplica

Como é possível observar, houve uma diferença significativa entre as métricas dos algoritmos investigados. Dentre os métodos **Pointwise**, o SVD Otimizado obteve a melhor acurácia e a maior diversidade, significativamente superior em relação aos métodos de Filtragem Baseada em Conteúdo. Além disso, é interessante notar que, nos algoritmos IAKNN, a similaridade baseada em apenas alguns atributos aumentou a diversidade, embora tenha diminuído o MAP.

Enquanto isso, na família **Pairwise**, comparou-se o impacto dos diferentes números de fatores no algoritmo BPRMF. Os dados coletados indicam que a acurácia baseada no MAP não se alterou de forma relevante, embora a diversidade tenha aumentado com um número de fatores maior. É interessante observar que os métodos Pointwise obtiveram uma precisão muito inferior ao dos métodos Pairwise e Listwise. Sabendo que quanto maior o ILS, menos diverso é o ranqueamento do algoritmo, esses métodos também obtiveram resultados menos diversos em comparação com os Pairwise e Listwise.

Na família **Listwise**, a acurácia obtida com o algoritmo ListNet foi muito superior às anteriores, embora não tenha sido possível coletar a métrica ILS devido à complexidade do modelo. Assim, é possível concluir que os algoritmos Pointwise, embora possuam uma implementação mais simples, apresentam resultados de acurácia e diversidade inferiores. Já os algoritmos Pairwise e Listwise, com uma implementação mais complexa, obtém resultados mais satisfatórios no problema em investigação.

REFERÊNCIAS

[1] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*. 129–136.

[2] Arthur da Costa, Eduardo Fressato, Fernando Neto, Marcelo Manzato, and Ricardo Campello. 2018. Case Recommender: A Flexible and Extensible Python Framework for Recommender Systems. In *Proceedings of the 12th ACM Conference on Recommender Systems* (Vancouver, British Columbia, Canada) *(RecSys '18)*. ACM, New York, NY, USA, 494–495. https://doi.org/10.1145/3240323.3241611

[3] Kim Falk. 2019. *Practical recommender systems*. Simon and Schuster.

[4] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[5] Hang Li. 2011. A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems* 94, 10 (2011), 1854–1862.

[6] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[7] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2022. Introduction to recommender systems handbook. In *Recommender systems handbook*. Springer, 1–35.

[8] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2000. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*. 158–167.

[9] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*. 22–32.

Received 06 Dezembro 2022