Marcos Antonio Tavares Galvão

Análise Comparativa de Sistemas de Recomendação de Animes Baseados em Filtragem Colaborativa Utilizando Dados do MyAnimeList

Universidade Federal de Pernambuco
graduacao@cin.ufpe.br
www.cin.ufpe.br/~graduacao

Recife
2024

Marcos Antonio Tavares Galvão

Análise Comparativa de Sistemas de Recomendação de Animes Baseados em Filtragem Colaborativa Utilizando Dados do MyAnimeList

Trabalho apresentado ao Programa de Graduação em Sistemas de Informação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Área de Concentração: *Inteligência Computacional*
Orientador: *Sérgio Ricardo de Melo Queiroz*

Recife
2024

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Galvão, Marcos Antonio Tavares.
Análise comparativa de sistemas de recomendação de animes baseados em filtragem colaborativa utilizando dados do MyAnimeList / Marcos Antonio Tavares Galvão. - Recife, 2024.
35p, tab.

Orientador(a): Sérgio Ricardo de Melo Queiroz
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Sistemas de Informação - Bacharelado, 2024.
Inclui referências, apêndices, anexos.

1. Inteligência Computacional. 2. Sistema de Recomendação. 3. Filtragem Colaborativa. 4. Aprendizado de Máquina. I. Queiroz, Sérgio Ricardo de Melo. (Orientação). II. Título.

000 CDD (22.ed.)

MARCOS ANTONIO TAVARES GALVÃO

ANÁLISE COMPARATIVA DE SISTEMAS DE RECOMENDAÇÃO DE ANIMES BASEADOS EM FILTRAGEM COLABORATIVA UTILIZANDO DADOS DO MYANIMELIST

Trabalho apresentado ao Programa de Graduação em Sistemas de Informação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação

Aprovado em: 02/08/2024

BANCA EXAMINADORA

_____

Prof. Sergio Ricardo de Melo Queiroz (Orientador)
Universidade Federal de Pernambuco

_____

Prof. Fernando Maciano de Paula Neto (Examinador Interno)
Universidade Federal de Pernambuco

Dedico este trabalho primeiramente aos meus pais, Doram e Graça, os quais acreditaram e me apoiaram, fazendo de tudo que lhes era possível para garantir minha educação e sustento. Em segundo lugar, gostaria de agradecer aos meus avós paternos, Maria e Sizedelo, por toda a contribuição para a minha educação e desenvolvimento.

Por fim, gostaria de agradecer aos meus amigos, aqueles que tornaram meu cotidiano especial. Aos que me fizeram feliz durante o ensino médio, aos que tanto me ensinaram na faculdade e aos que me motivaram no trabalho. Em especial, gostaria de agradecer a Guilherme, Igor e Arthur. Sem a contribuição, mesmo que indireta, de vocês, provavelmente este trabalho não estaria completo.

RESUMO

A popularização dos serviços de *streaming* facilitou o acesso a filmes e séries em um volume jamais visto. Nesse contexto, as animações japonesas (animes) apresentam um cenário muito desafiador para os usuários, que possuem uma gigantesca variedade de opções. Tal fato atrai o interesse por sistemas de recomendação, destinados à sugestão personalizada de conteúdo relevante. Sendo assim, este trabalho apresenta uma análise comparativa de quatro sistemas de recomendação baseados em filtragem colaborativa, utilizando os modelos Baseline, SVD, SVD++ e Co-Clustering, todos aplicados a dados da rede social MyAnimeList. O estudo apresenta a construção dos modelos, bem como a realização da otimização dos parâmetros através da técnica GridSearch. Por fim, é demonstrado como as diferentes abordagens se comportam sob as mesmas condições e exibe os resultados que mostram o modelo Baseline como o detentor da melhor métrica de F1-Score.

Palavras-chave: Sistemas de Recomendação, Filtragem Colaborativa, Aprendizado de Máquina

ABSTRACT

The popularization of streaming services has made it easier to access films and series in an unprecedented volume. In this context, Japanese animations (anime) present a challenging scenario for users, who have a huge variety of options. This is attracting interest in recommendation systems, designed to provide personalized suggestions of relevant content. This paper presents a comparative analysis of four recommendation systems based on collaborative filtering, using the Baseline, SVD, SVD++ and Co-Clustering models, all applied to data from the MyAnimeList social network. The study presents the construction of the models, as well as the optimization of the parameters using the GridSearch technique. Finally, it shows how the different approaches behave under the same conditions and displays the results that show the Baseline model as having the best F1-Score metric.

Keywords: Recommendation Systems, Collaborative Filtering, Machine Learning

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

FC	Sistemas de Recomendação Baseados em Filtro Colaborativo
MAL	MyAnimeList
RMSE	Raiz do Erro Quadrático Médio
SR	Sistema de Recomendação
SRs	Sistemas de Recomendação

SUMÁRIO

1

INTRODUÇÃO

Os filmes e séries são, indubitavelmente, alguns dos conteúdos de entretenimento mais consumidos em todo o mundo, gerando centenas de bilhões de dólares de lucro todos os anos. Segundo a Forbes [1] somente em 2019 a industria de filmes e séries já havia ultrapassado a marca de 100 bilhões de dólares arrecadados.

A Sony, por exemplo, adquiriu a Crunchyroll em 2020 [2], tornando-se dona da Funimation e da Crunchyroll, duas das maiores plataformas de *streaming* de animes da época. Atualmente a plataforma já possuí mais de 13 milhões de assinantes e a Sony estima que o mercado de animes alcançara cerca de $60 bilhões até 2030 [3]

Em um contexto tão diverso quanto o atual, com a disponibilidade de dezenas de serviços de *streaming* ao qual filmes e séries podem ser assistidos de casa, as animações japonesas, conhecidas como animes, não são uma exceção. Com uma vasta quantidade de obras e suas particularidades, escolher o que assistir se torna uma tarefa onerosa.

Nesse contexto, a aplicação de técnicas de aprendizado de máquina e de filtragem colaborativa tornaram-se fundamentais para o desenvolvimento de Sistemas de Recomendação. Estes, permitem a identificação de padrões de comportamento e similaridades entre os usuários, a fim de gerar recomendações relevantes e resultar em uma experiência mais agradável e simplificada [4].

A Netflix, que desempenhou e ainda desempenha um papel essencial na mudança de como as pessoas consomem entretenimento digitalmente [5], já em outubro 2006, iniciou o *The Netflix Prize*, uma competição para que os participantes desenvolvessem Sistemas de Recomendação Baseados em Filtro Colaborativo (FC) para sua plataforma e, por consequência, diversos avanços na área de Sistemas de Recomendação (SRs) ocorreram nos anos subsequentes [4].

Como demonstrado no trabalho de Yuan *et al.* [6], as técnicas de filtragem colaborativa são altamente aplicáveis para contextos envolvendo filmes e séries. Em seu trabalho, um FC utilizando-se de fatoração de matriz (discutiremos as técnicas SVD e SVD++ nas seções 3.1.2 e 3.1.3) foi aplicado aos dados do desafio do Netflix e do conjunto de dados MovieLens, apresentando um resultado que serve como base para trabalhos similares. A implementação do SVD++ (50D), por exemplo, foi capaz de alcançar um RMSE de 0.9401.

Trabalhos como Reynaldi *et al.* [7], por sua vez, se utilizam de técnicas de fácil implementação baseadas em memória para endereçar o mesmo desafio. Não se fazendo necessário o uso de modelos de aprendizado de máquina que aprenderam as preferencias dos usuários.

Já no trabalho de Soni *et al.* [8], foi proposto a criação de um SR híbrido (discutiremos na seção 2.1.3) utilizando técnicas de aprendizagem profunda e autocodificadores para produzir sugestões customizadas, o qual no conjunto de dados MovieLens atingiu um RMSE de 0.591. Já Reswara *et al.* [9], por sua vez, propôs outro sistema híbrido combinando a técnica de autocodificadores com similaridade por cosseno.

Em meio a tal cenário, este trabalho surge com o objetivo de realizar uma análise comparativa entre diferentes Sistemas de Recomendação Baseados em Filtro Colaborativo (FC) destinados a gerar recomendações customizadas de animes, utilizando-se de dados provenientes da rede social MyAnimeList (MAL) [10], a qual os usuários possuem a liberdade de criar suas próprias listas de animes, marcando e dando notas aos conteúdos que já assistiram.

Neste trabalho serão utilizadas diferentes técnicas de Filtragem Colaborativa, as quais terão suas qualidades avaliadas por meio de métricas como a Raiz do Erro Quadrático Médio (RMSE), Precisão, Revocação e F1-Score, tendo por fim, a solução escolhida sendo aplicada de maneira customizada a um usuário real.

Este trabalho visa contribuir para a área de sistemas de recomendação, oferecendo uma comparação entre as possíveis técnicas de criação de FC, mostrando como estas abordagens podem ser aplicadas para o contexto de animes, facilitando a descoberta de novas obras e aprimorando a experiência do usuário com base em suas preferências. Adicionalmente, estudos futuros podem ser baseados nos achados deste trabalho para avançarem ainda mais a qualidade da solução final.

2

SISTEMAS DE RECOMENDAÇÃO

Os Sistemas de Recomendação (SRs) possuem o desafiador objetivo de prover recomendações de itens de maneira adequada para usuários que nunca realizaram uma interação com o mesmo. Para atingir tal proposito o desafio pode ser abordado sob diversas óticas e se utilizar de diferentes técnicas para alcançar uma solução. A seguir, serão descritas algumas das técnicas mais populares de SRs e quais são os tipos de feedback que um SR pode utilizar.

2.1 TÉCNICAS DE RECOMENDAÇÃO

2.1.1 Recomendação baseado em Conteúdo

Os SRs baseados em conteúdo consistem em uma classe de algoritmos que realiza sugestões de itens aos usuários considerando as características, ou metadados, dos itens que o usuário já avaliou positivamente. Com isso o modelo pode inferir quais características levam a uma boa classificação para um determinado usuário e, com isso, busca mais itens similares para serem sugeridos [11].

De acordo com Ricci *et al.* [4] os SRs baseados em conteúdo seguem, em sua grande maioria, uma arquitetura de funcionamento dividida em 3 partes, sendo elas:

- Analisador de Conteúdo: Caso a informação advinda do item analisado não esteja de modo estruturado, será necessário algum tipo de pré-processamento. Esta informação pode ser oriunda de uma análise realizada com outros algoritmos ou através de uma representação alternativa para que seja utilizada pela próxima parte do sistema.
- Aprendiz de Perfil: Esta parte se caracteriza pela aplicação de técnicas de aprendizado de máquina para inferir quais características dos itens associados ao usuário são relevantes para serem considerados em futuras recomendação.
- Componente de Filtragem: Neste módulo o Aprendiz de Perfil será aplicado para encontrar itens relevantes e uma seleção dos mais relevantes produzirá o resultado final.

2.1.2 Recomendação baseada em Filtro Colaborativo

O FC é historicamente uma das técnicas mais bem-sucedidas de SRs. Consistem em uma classe de técnicas que se utiliza da existência de feedbacks de vários usuários para produzir novas recomendações [12].

Estes feedbacks considerados pelo FC podem ser de origem explícita (vide seção 2.2.1), como notas fornecidas pelos próprios usuários para animes no MyAnimeList (MAL), ou implícitas para cenários aos quais a inferência do feedback se fez necessária, vide seção 2.2.2 [4].

Assim como descrito por Su e Khoshgoftaar [12] os Sistemas de Recomendação Baseados em Filtro Colaborativo podem ser divididos em 3 categorias:

- FC Baseado em Memória: Compreende os algorítimos de vizinhança ou que utilizam matrizes de correlação.
- FC Baseado em Modelo: Compreende os modelos de Clusterização (CoClustering descrito na seção 3.1.4) , algoritmos de redução de dimensionalidade (SVD e SVD++, descritos respectivamente nas seções 3.1.2 e 3.1.3), Redes Bayesianas, etc.
- Recomendadores Híbridos: Serão descritos na seção 2.1.3.

Ainda como apresentado por Su e Khoshgoftaar [12], cada categoria possui suas vantagens e desvantagens. Focando apenas nas duas primeiras categorias, visto que os Recomendadores Híbridos serão explorados separadamente, podem ser citados como principais pontos de força e de fraqueza:

- FC Baseado em Memória:
  - Vantagens: Fácil implementação e novos dados podem ser facilmente adicionados.
  - Desvantagens: A qualidade despenca quando os dados são esparsos, não é possível recomendar para novos usuários.
- FC Baseado em Modelo:
  - Vantagens: Lida melhor com dados esparsos e conseguem atingir performances melhores.
  - Desvantagens: Possui uma estrutura mais complexa e geralmente apresenta dificuldades de escalabilidade mantendo a performance.

Embora existam diferentes abordagens para os FCs, seus objetivos permanecem consistentes, sendo, em linhas gerais, a representação de um indivíduo com gosto similar a um novo usuário, buscando realizar sugestões (recomendações) de itens promissores [13].

2.1.3 Recomendação baseada em abordagem Híbrida

Com a conceituação dos Sistemas de Recomendação Baseados em Filtro Colaborativo (FC) e dos Sistemas de Recomendação (SRs) Baseados em Conteúdo, fica claro suas diferenças de abordagem, onde ambas as técnicas observam o problema sob diferentes perspectivas.

Por consequência, considerando tais fatos, os Sistemas de Recomendação (SRs) Híbrido ganham espaço de atuação, apresentando uma abordagem de mescla de diversos SRs podendo eles ser da mesma categoria ou não [4].

Devido as qualidades e fraquezas individuais de cada técnica de SRs, a combinação delas busca, em suma, obter resultados mais eficientes e que possam cobrir cenários que individualmente são mais desafiadores [14].

Segundo Burke [15], existem 7 formas de classificar as diferentes formas de combinar técnicas de recomendação individuais em um SR híbrido, são elas:

- *Weigthed* (Híbrido Ponderado): Nesse método, as saídas dos SRs são ponderadas para gerar uma única recomendação. A ponderação atribuída a cada técnica pode ser fixa ou dinamicamente ajustada com base no feedback do usuário que receberá a recomendação [16].

- *Switching* (Híbrido de Chaveamento): É uma técnica utilizada que altera entre os SRs com base em algum critério. Por exemplo, para novos usuários é utilizado um SR baseado em conteúdo e para os demais cenários um FC é escolhido [4].

- *Mixed* (Híbrido Misto): As recomendações advindas de diferentes técnicas são apresentadas simultaneamente ao usuário, sem combinação das pontuações. É útil para cenários aos quais múltiplas recomendações simultâneas são fornecidas.

- *Feature combination* (Combinação de Características): Características observadas por um algoritmo são combinadas e se tornam características adicionais dos dados observados pela técnica seguinte. Por exemplo, tornar a interação dos usuários observados por um FC em uma nova característica que será repassada para um SRs baseado em conteúdo [4].

- *Cascade* (Cascata): As recomendações avançam em cascata entre as técnicas, onde a anterior filtra as informações que chegarão na técnica seguinte.

- *Feature augmentation* (Aumento de Características): A saída de uma técnica de recomendação é adicionada aos dados. Por exemplo, observações geradas por um classificador de conteúdo podem ser usadas como características adicionais para os dados de um modelo colaborativo [16].

- *Meta-level* (Meta-Nível): Nessa abordagem as saídas diretas de uma técnica se tornam as entradas de outra [17]. Por exemplo, um FC é utilizado para produzir um

perfil para cada usuário, e partindo deste perfil um SR baseado em conteúdo produz as recomendações [16].

Embora os SR híbrido possuem um grande potencial de desempenho, visto que na maioria das vezes sistemas híbridos produzem melhores resultados que sistemas individuais [12], sua implementação vem com uma série de desafios, Su e Khoshgoftaar [12] citam como sendo alguns deles, o aumento de complexidade de implementação e a utilização de informações adicionais, visto que os dados de feedback dos usuários não se faz suficiente.

2.2 TIPOS DE FEEDBACK

Os feedbacks consistem em como o usuário responde a um dado item, eles podem ser divididos em duas categorias, sendo elas descritas abaixo.

2.2.1 Feedback Explícito

Se trata da opinião explicitamente fornecida pelo usuário, por exemplo, qual nota o usuário A forneceu para o anime B. Normalmente, classificações deste tipo são as mais usuais para os SRs [4].

Segundo apresentado por Schafer *et al.* [18] essa forma de feedback (nota) pode se apresentar das seguintes formas:

- Nota Escalar: Pode se tratar de uma nota numérica, por exemplo de 1 a 5, ou de forma ordinal, por exemplo, concordo fortemente, discordo, etc.
- Binário: Gosto ou não gosto, concordo ou não concordo, etc.
- Unário: Quando presente, mapeia a interação do usuário com o item e em caso de ausência informa a inexistência dela.

2.2.2 Feedback Implícito

O feedback implícito consiste em inferir o feedback de um usuário por meio de seu comportamento com os itens [19]. A relevância de tal técnica advém da grande dificuldade de obter dados de feedback explícito em um volume suficiente para evitar esparcidade das interações usuário-item [20]. Em cenários como este, o feedback implícito adiciona relevante informação para melhor tomada de decisão dos modelos.

3

ALGORITMOS DE RECOMENDAÇÃO

3.1 ALGORITMOS

Nesta seção, serão descritos os principais algoritmos testados no conjunto de dados do MAL, visando, por fim, a escolha do modelo mais adequada. O processo detalhado de escolha está localizado na seção 4.

As estratégias a seguir pertencem a três categorias distintas: Baseline (Linha de Base), Fatoração de Matriz e Clusterização.

3.1.1 Baseline

Os FC buscam mapear a interação entre usuários e itens, contudo, muitos dos fatores determinantes para esse mapeamento são intrínsecos ao usuário ou aos itens [4]. Partindo deste conceito, os previsores baseados na técnica de *Baseline* buscam aprender quais fatores, que serão chamados de vieses, são particulares de cada usuário e de cada item e, com isso, utilizar-se desse conhecimento para estimar a nota de um usuário para um item.

A ideia do algoritmo parte da noção que um usuário tende a ter uma média de notas diferente da média global de todos os itens, seja porque o usuário é mais criterioso, fornecendo notas mais baixas que a média dos outros usuários, ou porque ele costuma normalmente dar notas mais altas que os demais. Isto, portanto, definiria o viés deste usuário, ou seja, o quão distante da média global ele costuma classificar os seus itens [4].

De mesmo modo, tal circunstância também acontece com os itens. Um dado item, digamos que muito popular, costuma ser melhor avaliado que a média global das notas dos itens. Em contrapartida, um item muito impopular é mais frequentemente pior avaliado que a média global dos itens. Esse desvio será caracterizado como o viés deste item em específico [4].

Com isso, definindo os termos μ como sendo a média geral das notas, b_u como sendo o viés de nota do usuário, b_i como sendo o viés de nota do item. A nota de um usuário u para um item i é denotado como r_{ui} e a estimativa de nota como $\hat{r}_{ui}$. A formula a seguir descreve a combinação de tais termos:

$$\hat{r}_{ui} = \mu + b_u + b_i \tag{3.1}$$

Os valores para os parâmetros b_u e b_i são encontrados através da resolução dos mínimos quadrados denotados pela fórmula:

$$\min_{b*} \sum_{(u,i)\in K} (r_{ui} - \mu - b_u - b_i)^2 + \lambda(\sum_u b_u^2 + \sum_i b_i^2) \tag{3.2}$$

A primeira parte da equação $\sum_{(u,i)\in K}(r_{ui} - \mu - b_u - b_i)^2$ busca encontrar os valores de b_u e b_i que atendam uma dada nota r_{ui}. O segundo termo $\sum_u b_u^2 + \sum_i b_i^2$ age como regularizador evitando *overfitting* e punindo parâmetros com altos valores.

O parâmetro λ se trata de quão intensa a regularização da equação será para a magnitude dos parâmetros [4].

A solução para equação 3.2 pode ser alcançada através de Gradiente Descendente (Stochastic Gradient Descent) [4].

3.1.2 SVD

O SVD consiste em uma técnica de fatoração de matriz. Tais técnicas se demonstraram capazes de encontrar características escondidas das interações usuário-item que possam explicar as motivações por trás dos feedbacks [4].

Assim como sugerido por Marlin *et al.* [21], a utilização dos dados de interação usuário-item explícitas e implícitas, juntamente com uma boa adequação de regularização do modelo, podem gerar resultados mais precisos.

Os modelos de SVD sofrem caso treinados em conjuntos de dados esparsos, situação encontrada na maioria dos cenários de FCs, pois a maioria dos usuários interagem apenas com um pequeno subconjunto do total de itens existentes. Ocasionando, em grandes vazios de interação usuário-item [4], portanto, fazem-se necessárias maneiras de endereçar tais situações.

Tendo isso em mente os FC que utilizam SVD localizam essas características escondidas, denotadas pelo vetor $q_i \in \mathbb{R}^f$, que contem o quão presente uma característica está para o item i. Já o vetor $p_u \in \mathbb{R}^f$, denota o quão interessado em cada uma destas características um usuário u está. Com isso, o fator regulador gerado pelo SVD será definido como o seguinte produto escalar:

$$q_i^T p_u \tag{3.3}$$

A estimativa final de nota será gerada através da formula conjunta de um modelo de baseline (3.1) integrado com o termo regularizador (3.3), sendo ela:

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u \tag{3.4}$$

Os parâmetros b_i, b_u, q_i e p_u serão encontrados através da resolução da formula dos mínimos quadrados regularizados:

$$\min_{b_*,q_*,p_*} \sum_{(u,i)\in K} (r_{ui} - \mu - b_i - b_u - q_i^T p_u)^2 + \lambda(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2) \tag{3.5}$$

Assim como a solução para equação 3.2, a equação 3.5 pode ser resolvida através de Gradiente Descendente (Stochastic Gradient Descent) [22].

O parâmetro λ se trata de quão intensa a regularização da equação será aplicada, considerando a magnitude das características escondidas encontradas pelo SVD [4].

3.1.3 SVD++

Os modelos que se utilizam da técnica chamada de SVD++ [23], consistem de uma versão aprimorada do SVD tradicional (3.1.2), buscando tomar melhor proveito do feedback implícito existente nas interações usuário-item. Tal objetivo é alcançado através de uma representação mais complexa da perspectiva do usuário, que se trata do termo p_u da equação de regularização apresentada em 3.3 [4].

Para o SVD++ o novo termo de regularização torna-se [23]:

$$q_i^T \left(p_u + |R(u)|^{-\frac{1}{2}} \sum_{j\in R(u)} y_j \right) \tag{3.6}$$

O termo regulador ainda mantém as variáveis q_i^T e p_u assim como em 3.3, porém, o novo termo $|R(u)|^{-\frac{1}{2}} \sum_{j\in R(u)} y_j$ foi adicionado, o qual representa a perspectiva do usuário (notas) normalizada. O termo $R(u)$ possui as notas dos itens aos quais o usuário u interagiu [4].

Portanto, a equação final para estimar a nota do usuário u para o item i é dada por:

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T \left(p_u + |R(u)|^{-\frac{1}{2}} \sum_{j\in R(u)} y_j \right) \tag{3.7}$$

Assim como o SVD tradicional, os parâmetros b_i, b_u, q_i e p_u serão encontrados através de Gradiente Descendente.

3.1.4 Co-Clustering

A implementação que será utilizada como base de explicação para o Co-Clustering [24] a seguir, surge com o objetivo de ser de rápido treinamento, funcional com novos usuários e itens sem a necessidade de retreino completo do modelo e com precisão similar aos algoritmos baseline e de fatoração de matriz.

A técnica consiste em um algoritmo de clusterização baseado no algoritmo de co-clustering ponderado proposto por Banerjee *et al.* [25], que endereça clusters de usuários e itens simultaneamente [24], utilizando a média de notas dos vizinhos (objetos de um mesmo cluster) para estimar uma nota para uma interação que ainda não ocorreu [26].

Como veremos a seguir, a matriz A corresponde a relação dos usuários com os itens, sendo A_{ij} a nota do usuário i para o item j. Por sua vez, a matriz W_{ij} denota se houve a interação entre o usuário i e o item j.

Já o termo $\hat{A}_{ij}$, denota a estimativa de nota para o usuário i e o item j, sendo ela calculada pela formula a seguir:

$$\hat{A}_{ij} = A^{CCC}_{gh} + (A^{R}_{i} - A^{RC}_{g}) + (A^{C}_{j} - A^{CC}_{h}) \tag{3.8}$$

O primeiro termo em 3.8, denotado por A^{CCC}_{gh} se trata da média da nota do co-cluster (global). O segundo termo $(A^{R}_{i} - A^{RC}_{g})$, se trata da subtração da média de nota do usuário e a média de nota do cluster do usuário. O terceiro termo $(A^{C}_{j} - A^{CC}_{h})$, se trata da subtração da média de nota do item entre todos os usuários e a média de nota do cluster do item [24].

O processo de encontrar os clusteres, definido como p para usuários e γ para itens, é otimizado através da resolução dos mínimos quadrados da equação:

$$\min_{(p,\gamma)} \sum_{i=1}^{m} \sum_{j=1}^{n} W_{ij}(A_{ij} - \hat{A}_{ij})^2 \tag{3.9}$$

Uma vez treinado, o algoritmo de Co-Clustering será capaz de estimar a nota que um usuário dará a um item através da obtenção de seus clusters e a resolução da formula 3.8.

3.2 MÉTRICAS DE AVALIAÇÃO

A qualidade de um Sistema de Recomendação (SR) é medido por métricas que avaliam o quão bem o sistema está performando, bem como o quão satisfatória as recomendações são para os usuários. As métricas mais relevantes para o trabalho são descritas a seguir.

3.2.1 Raiz do Erro Quadrático Médio

O RMSE, sigla em inglês para *Root Mean Square Error* (Raiz do Erro Quadrático Médio), mede a diferença entre as notas previstas pelo SR e as notas reais dadas pelos usuários. Quanto menor o RMSE, melhor a performance do sistema, pois indica que as previsões estão mais próximas dos valores reais. Ele é calculado da seguinte forma:

$$RMSE = \sqrt{\frac{1}{N} \sum_{u,i} (r_{ui} - \hat{r}_{ui})^2}$$

Onde:

- r_{ui} é a nota real dada pelo usuário u ao item i
- $\hat{r}_{ui}$ é a nota prevista pelo sistema para o usuário u e o item i
- N é o número total de notas

3.2.2 Precisão

A precisão mede a proporção de itens recomendados que são relevantes para o usuário. Em outras palavras, avalia o quão capaz é o sistema de recomendar itens que o usuário considere relevante. É calculada da seguinte forma:

$$\text{Precisão} = \frac{\text{Número de itens recomendados relevantes}}{\text{Número total de itens recomendados}}$$

3.2.3 Revocação

A revocação mede a proporção dos itens relevantes para o usuário que foram recomendados pelo sistema. Diferente da precisão, a revocação avalia a capacidade do sistema de não deixar de recomendar itens que o usuário provavelmente se interessaria. É calculado pela seguinte fórmula:

$$\text{Revocação} = \frac{\text{Número de itens recomendados relevantes}}{\text{Número total de itens relevantes}}$$

3.2.4 F1-Score

O F1-Score é a média harmônica entre precisão e revocação. Ele é útil para equilibrar a importância de ambas as métricas, especialmente em situações onde há um trade-off entre elas. Um valor alto de F1-Score indica que o sistema tem bom desempenho tanto em recomendar quanto em não omitir itens relevantes. É calculado da seguinte forma:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precisão} \cdot \text{Revocação}}{\text{Precisão} + \text{Revocação}}$$

4

METODOLOGIA

Neste capítulo, serão abordados os detalhamentos do conjunto de dados e quais dados serão utilizados pelos modelos. Por fim, a explicação de todos os experimentos realizados, bem como o Sistema de Recomendação baseado em Filtro Colaborativo resultante do trabalho.

4.1 CONJUNTO DE DADOS

Para este trabalho, o conjunto de dados utilizados advém da rede social destinada ao público de animes, o MyAnimeList (MAL). O conjunto de dados foi disponibilizado na plataforma online Kaggle, pelo seu autor Matěj Račinský [27].

O conjunto original é composto de três arquivos:

1. Dados dos Anime: Contém informação sobre 6.668 animes cadastrados na plataforma MAL.

2. Dados dos Usuários: Contém informações públicas de 108.711 usuários da plataforma.

3. Dados das Notas: Contém as interações usuário-anime. Aqui estão contidos quais animes cada usuário assistiu e quais notas eles deram. Este conjunto de dados possuí 1.558.384 registros.

Embora existam três conjuntos de dados, o mais relevante para o trabalho será o conjunto 3, isso porque é nele que os feedbacks explícitos (2.2.1) de interação dos usuários com os itens estão localizados. Os demais conjuntos de dados, 1 e 2, servirão apenas para enriquecer as informações saídas do SR, isto porque a listagem de notas apenas guarda os identificadores dos usuários e animes, logo, os demais conjuntos fornecem qual o nome do anime e qual o nome do usuário. Na tabela 1 estão expressas quais colunas (*features*) de cada conjunto de dados são utilizadas.

Tabela 1: Colunas (*features*) utilizadas do conjunto de dados do MyAnimeList

Conjunto	Colunas
Animes	Identificador e Título
Usuário	Identificador e Nome de usuário
Notas	Identificador do usuário, Identificador do anime e Nota

4.2 EXPLORAÇÃO DO CONJUNTO DE DADOS

A exploração dos dados será concentrada no Conjunto das Notas (seção 4.1 conjunto 3). Primeiramente, é importante salientar que, na plataforma MAL, todas as notas se encontram entre 1 e 10, sendo respectivamente, a mais baixa e a mais alta. Não existem notas 0 na plataforma, sendo tal valor representante de interações que não receberam uma nota. Portanto, daqui em diante, toda a tratativa considera apenas interações com notas acima de 0.

O conjunto de dados original possui registros de quais animes o usuário ainda está assistindo ou quais ele planeja assistir, porém, para o contexto aqui aplicado serão considerados apenas os animes marcados como completados (totalmente assistidos) pelo usuário.

Existem ao todo 1.558.384 registros de interação; após a remoção daquelas com nota 0, restaram 548.653. Cada registro possui três informações. Um identificador de qual usuário o registro pertence, o identificador de qual anime se trata e qual a nota o usuário atribuiu.

Dos 108.711 usuários do conjunto de usuários (seção 4.1 conjunto 2), apenas 77.563 completaram algum anime, sendo este segundo número a quantidade de usuários com registros de interação. E dos 6.668 animes (seção 4.1 conjunto 1) apenas 5.034 possuem avaliações.

Tabela 2: Animes, Usuários e Interação após desconsiderar notas 0

Conjunto	Total	Restantes	Diferença
Animes	6668	5034	-1634
Usuários	108711	77563	-31.148
Interações	1558384	548653	-1.009.731

Abaixo segue a distribuição da quantidade de animes assistidos pelos usuários. É possível observar que a grande parte se localiza muito abaixo dos 100 animes completos.

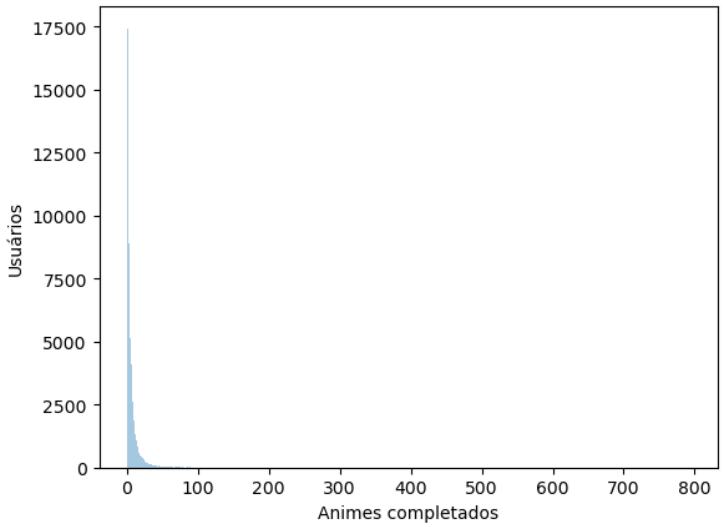

Figura 1: Distribuição do número de animes assistidos por usuário

Tabela 3: Distribuição estatística do número de animes assistidos por usuário

Média	7.07
Desvio padrão	12.94
Min	1
25%	2
50%	4
75%	8
Max	793

A partir das informações estatísticas apresentadas na Tabela 3, é perceptível que nem todos os usuários assistiram a todos os animes, o que acarreta em um conjunto de dados esparso. Para este conjunto de dados existe uma esparsidade de 97.35% [4].

Já a distribuição das notas pode ser visualizada na tabela 4, onde é notável que a média de notas se encontra próximo de 8 (sendo 7.99) e que a grande maioria das notas são maiores que 7, caracterizando um conjunto de dados desbalanceado [4].

Tabela 4: Distribuição estatística das notas

Total de Notas	548653
Média	7.99
Desvio padrão	1.49
Min	1
25%	7
50%	8
75%	9
Max	10

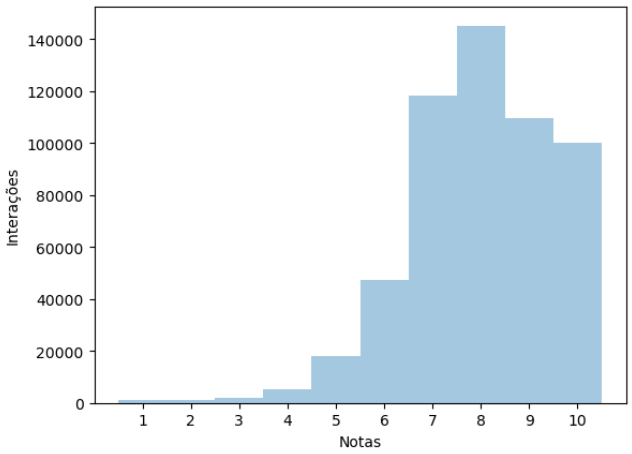

Figura 2: Distribuição do número de notas

Assim como descrito na seção 3, cada técnica de filtragem colaborativa possui suas vantagens e desvantagens, bem como conseguem endereçar o problema sob vieses distintos. Com isso em mente, a seção seguinte (4.3) tratará de testar tais técnicas com os dados em mãos.

4.3 EXPERIMENTOS

Para os experimentos de criação dos FCs foi escolhida a linguagem de programação Python [28] por conta de sua versatilidade e disponibilidade de bibliotecas para manipulação de dados. A biblioteca Surprise [29] foi escolhida para auxiliar na criação dos modelos, já que ela contem diversas implementações voltadas para criação de FC. Ademais, as bibliotecas Numpy [30] e Pandas [31] foram utilizadas para manipulação dos dados.

Todos os testes foram realizados em uma máquina equipada com um processador Ryzen 7 7700X e 32gb de memória RAM DDR5.

Os seguintes algoritmos da biblioteca Surprise foram escolhidos para serem testados no conjunto de dados, são eles:

- *BaselineOnly*, sendo esse uma implementação do algoritmo descrito na seção 3.1.1.

- *SVD*, sendo esse uma implementação do algoritmo descrito na seção 3.1.2.
- *SVD++*, sendo esse uma implementação do algoritmo descrito na seção 3.1.3.
- *CoClustering*, sendo esse uma implementação do algoritmo descrito na seção 3.1.4.

Os quatro modelos foram treinados seguindo uma divisão de 75% dos dados para treinamento e 25% para validação. Esses dados são compostos de notas atribuídas por usuários apenas a animes que eles marcaram como assistido (no MAL, marcados como '*Completed*'). Durante todo o processo de treinamento e validação, o estado aleatório do Python foi atribuído para o valor arbitrário 55, a fim de evitar distinções entre execuções.

As quatro métricas explicadas na seção 3.2 foram extraídas após a execução do treinamento nos dados de validação, sendo elas:

- *Raiz do Erro Quadrático Médio*, descrito na seção 3.2.1
- *Precisão*, descrito na seção 3.2.2
- *Revocação*, descrito na seção 3.2.3
- *F1-Score*, descrito na seção 3.2.4

As métricas RMSE e F1-Score serão utilizadas para seleção final dos modelos. Sendo o RMSE para esclarecer o quão longe dos *feedbacks* reais as previsões estão e o F1-Score para mensurar de maneira "harmônica"o balanço entre Precisão e Revocação, visto que o conjunto de dados faz-se desafiador para ambas as métricas [4].

O treinamento dos modelos se deu em paralelo, todos utilizando os parâmetros padrões do Surprise, vide tabela 5, advindos dos artigos de referência de suas implementações [29].

Tabela 5: Valores dos parâmetros utilizados em cada modelo na fase de treinamento

Parâmetro	Baseline	SVD	SVD++	Co-Clustering
Épocas de treinamento	20	20	20	20
Taxa de aprendizado	0.005	0.005	0.007	
Taxa de regularização	0.03	0.02	0.02	
Número de *clusters* de usuários				3
Número de *clusters* de itens				3

As métricas extraídas correspondem ao TOP-N com $N = 10$, ou seja, as estimativas produzidas pelos modelos são ordenadas de maneira decrescente e somente as 10 maiores estimativas são utilizadas para computar as métricas de avaliação. Portando, se tratam do RMSE@10, Precisão@10, Revocação@10 e F1-Score@10.

O TOP-10 foi escolhido, pois, embora os modelos sejam capazes de estimar a relevância de todos os animes para o usuário, apenas um subconjunto dessas sugestões é realmente relevante,

sendo as demais estimativas imprecisas, portando grandes *N*s implicam em menores F1-Scores e maiores desvios padrões sendo observados nas métricas [4].

Tabela 6: Métricas obtidas nos dados de validação

Modelo	RMSE@10	Precisão@10	Revocação@10	F1-Score@10	Tempo para treinar
BaselineOnly	1.227847	0.894500	0.973276	0.895683	1.75 segs
SVD	1.223144	0.901477	0.955143	0.888263	3.59 segs
SVD++	1.212198	0.907860	0.938665	0.880179	12.82 segs
CoClustering	1.281609	0.924979	0.882388	0.839734	7.01 segs

Os desvios padrões das métricas Precisão, Revocação e F1-Score foram calculados e estão descritos na tabela a seguir:

Tabela 7: Desvio padrão das métricas obtidas nos dados de validação

Modelo	Std Precisão@10	Std Revocação@10	Std F1-Score@10
BaselineOnly	0.257010	0.125153	0.254091
SVD	0.251227	0.165906	0.262929
SVD++	0.245535	0.197375	0.273029
CoClustering	0.223449	0.287157	0.321850

Após a execução da fase de treinamento e extração de métricas no conjunto de validação, observou-se que o melhor modelo foi o BaselineOnly (3.1.1). Embora seja a implementação mais simples dentre as testadas, foi a alternativa que melhor conseguiu sugerir itens aos usuários. O mesmo, obteve o maior F1-Score, sendo ele de aproximadamente 0.896, e obteve o menor desvio padrão nesta métrica, sendo ele de 0.254. Embora ambas as métricas tenham sido apenas levemente superiores aos demais modelos, seu tempo de treino de apenas 1.75 segundos contribui para seu uso em cenários com novos itens ou usuários, o qual requer de re-treino do modelo [4].

Para tentar extrair ainda mais desempenho dos modelos, realizou-se uma busca de parâmetros exaustiva utilizando a técnica *GridSearch* do Surprise [29], a qual é baseada na implementação do SciKitLearn [32].

Os parâmetros escolhidos para cada modelo foram baseados nos seus artigos de referência, incluindo valores adicionais com pequenos incrementos. A validação dos parâmetros foi realizada utilizando *cross-validation* (Validação Cruzada) em 5 subconjuntos [33].

Os parâmetros em que a busca exaustiva (GridSearch) atuou estão especificados nas tabelas 8, 9 e 10, sendo respectivamente destinadas ao BaselineOnly, SVD e SVD++, e Co-Clustering.

Tabela 8: Valores utilizados na busca de parâmetros para o modelo *BaselineOnly*

Parâmetro	Valores testados
Épocas de treinamento	20, 50, 250, 500, 1000
Taxa de aprendizado	0.005, 0.007, 0.01
Taxa de regularização	0.01, 0.03, 0.1

Tabela 9: Valores utilizados na busca de parâmetros para os modelos SVD e SVD++

Parâmetro	Valores testados
Épocas de treinamento	10, 20, 50, 100, 250, 1000
Taxa de aprendizado	0.001, 0.005, 0.007, 0.01
Taxa de regularização	0.02, 0.015

Tabela 10: Valores utilizados na busca de parâmetros para o modelo Co-Clustering

Parâmetro	Valores testados
Épocas de treinamento	10, 20, 50, 100, 500, 1000
Número de *clusters* de usuários	3, 5, 7, 11
Número de *clusters* de itens	3, 5, 7, 11

Tabela 11: Valores dos parâmetros de cada modelo após otimização via GridSearch

Parâmetro	Baseline	SVD	SVD++	CoClustering
Épocas de treinamento	20	1000	100	20
Taxa de aprendizado	0.005	0.007	0.001	
Taxa de regularização	0.03	0.02	0.02	
Número de *clusters* de usuários				3
Número de *clusters* de itens				3

Após a otimização de parâmetros, a melhor configuração para cada modelo foi re-treinada no conjunto de treinamento e validada do conjunto de validação. Na tabela 11 é possível visualizar quais foram os melhores parâmetros encontrados para cada um dos modelos.

Tabela 12: Métricas de avaliação após Busca de Parâmetros

Modelo	RMSE@10	F1-Score@10	Std F1-Score	Tempo para treinar
BaselineOnly	1.227847	0.895683	0.254091	2.38 segs
SVD++	1.202079	0.890355	0.261066	56.35 segs
SVD	1.207276	0.889986	0.260768	139.58 segs
CoClustering	1.281606	0.839736	0.321849	7.31 segs

Foi observada uma melhora mínima nos F1-Scores dos modelos (vide tabela 12), tendo o BaselineOnly permanecido o mesmo. Porém, foi observado um menor RMSE para os modelos SVD e SVD++ potencializado pelo maior número de épocas do gradiente descendente. Embora o modelo baseado em SVD++ tenha superado sua versão simplificada, o SVD, ainda não foi o suficiente para obter melhor F1-Score.

Tabela 13: Diferença nas métricas de avaliação após a Busca de Parâmetros

Modelo	RMSE@10	F1-Score@10	Std F1-Score	Tempo para treinar
BaselineOnly	0	0	0	+ 0.58 segs
SVD++	- 0.01	+ 0.01	- 0.012	+ 43.36 segs
SVD	- 0.0158	+ 0.0017	- 0.002	+ 135.47 segs
CoClustering	- 0.000003	+ 0.000002	-0.000001	+ 0.20 segs

Por fim, o modelo final escolhido foi o BaselineOnly, que demonstrou a melhor métrica F1-Score e o menor tempo necessário para treino, também apresentando um RMSE bem próximo aos demais modelos. Ademais, todos os modelos utilizaram-se da técnica de Gradiente Descendente para encontrar seus parâmetros.

5

NOVOS USUÁRIOS

Se tratando de SRs, a geração de recomendações para novos usuários é um desafio recorrente para aplicações que lidam com cenários dinâmicos, onde a quantidade de usuários pode aumentar com frequência [4].

Por exemplo, uma vez o modelo resultante da seção 4.3 estando treinado, ele é capaz de gerar recomendações, calculando as estimativas de notas, para todos os usuários que foram observados no conjunto de dados de treino. Porém, caso um usuário que não estava presente na fase de treino seja apresentado, o modelo não será capaz de realizar sugestões que considerem as preferências deste indivíduo.

Tal fato pode ser observado pela formula 3.1 do modelo *Baseline*. O termo b_u, que representa o viés do usuário, não seria conhecido pelo modelo, já que o usuário era inexistente na fase de treinamento. Portanto, a estimativa das notas para este novo usuário seria resultante da soma de apenas parte dos termos, sendo eles, a média global (μ) e o viés do item (b_i).

Sendo assim, para aplicar o modelo para um novo usuário é necessário, primeiramente, que a listagem de feedbacks explícitos (notas) deste usuário seja apresentada ao modelo, para que a inferência das suas preferências seja realizada. A seguir é descrito o passo a passo para endereçar este problema:

1. Coletar os *feedbacks* explícitos do novo usuário;
2. Realizar a concatenação dos conjuntos de dados das interações usuário-item (seção 4.1 conjunto 3) e dos registros de quais animes foram assistidos (com as notas) pelo novo usuário;
3. Realizar o re-treino do modelo nesse novo conjunto completo dos dados;
4. Calcular estimativas e gerar as recomendações.

A seguir (figura 3) é apresentado um exemplo de uma recomendação para um novo usuário que não estava presente no conjunto de dados treinamento da seção 4.3.

	rec	title	Usuário: Galtvam
1	8.57	Gintama': Enchousen	
2	8.48	Gintama°	
3	8.42	Gintama'	
4	8.42	One Piece	
5	8.38	Gintama.	
6	8.29	Gintama.: Shirogane no Tamashii-hen	
7	8.27	Gintama.: Porori-hen	
8	8.14	Pingu in the City	
9	8.12	Hunter x Hunter (2011)	
10	8.11	Hajime no Ippo: New Challenger	
11	8.08	Detective Conan	
12	8.07	Gintama	
13	8.05	Uchuu Kyoudai	
14	8.04	Saiki Kusuo no Ψ-nan (TV) 2	
15	8.03	Monogatari Series: Second Season	

Figura 3: Exemplo de recomendação para um novo usuário

6

AVALIANDO RESULTADO

O Sistema de Recomendação baseado em Filtro Colaborativo resultante da fase de treinamento e otimização, descritas na seção 4.3, consiste em um algoritmo de linha de base (*Baseline*) descrito na seção 3.1.1.

Os parâmetros do modelo foram definidos como 50 épocas de treinamento para o gradiente descendente, uma taxa de aprendizagem de *0.005* e uma taxa de regularização de *0.03*.

O modelo foi treinado em uma máquina equipada com um processador Ryzen 7 7700X com 32gb de memória RAM, se mostrando capaz de ser treinado em 548.653 registros (vide tabela 2) em 1.75 e 2.38 segundos. Tal característica se torna útil para cenários onde novos usuários ou novos animes precisam ser incluídos na base de dados, o que implica na necessidade do re-treino do modelo [4].

O modelo alcançou um valor para a Raiz do Erro Quadrático Médio (RMSE) de *1.227847* e um F1-Score de *0.895683*, apresentando um desvio padrão de *0.254091* nesta mesma métrica.

Embora não seja uma comparação direta, a implementação do FC feito por Yuan *et al.* [6] do SVD++ (50D) aplicado no conjunto de dados ML-100k (destinado a filmes) obteve um RMSE de *0.9401*. O modelo PDMF-post, do mesmo trabalho, obteve por sua vez *0.6538*. Em comparação, o sistema híbrido para animes RikoNet [8], ainda aplicado ao conjunto de dados ML-100K, obteve um RMSE de *0.591*.

Em comparação com as métricas apresentadas acima, é possível inferir que é muito provável a possibilidade de extrair ainda mais desempenho do conjunto de dados utilizado. Seja através de otimizações no modelo atual, ou partindo para uma abordagem de sistemas híbridos que, muito certamente, conseguiria produzir resultados ainda mais promissores.

7

CONCLUSÃO

Neste trabalho, foi realizado uma análise comparativa de diversas abordagens para a produção de Sistemas de Recomendação, tendo como proposito avaliar qual modelo baseado na técnica de Filtragem Colaborativo foi capaz de realizar sugestões para usuários com melhor qualidade. Tudo isso baseado-se no registro de animes assistidos por cada indivíduo. Como base de dados foi utilizado o conjunto de dados dos usuários do MyAnimeList.

Neste trabalho, inicialmente, foi realizado uma contextualização sobre o que são os sistemas de recomendação e quais abordagens existem (2), sendo elas a Baseada em Conteúdo (2.1.1), Filtro Colaborativo (2.1.2) e Híbrido (2.1.3). Após, foi falado sobre alguns dos algoritmos mais populares para produção de filtros colaborativos, os quais foram utilizados no trabalho.

Após a contextualização, uma análise da situação dos dados de interação usuário-anime foi realizada. Demonstrando as características do conjunto aos quais os modelos seriam submetidos. Posteriormente, a seção de experimentos foi descrita. Passando primariamente pela fase de treinamento e validação de todos os modelos e, em seguida, realizando uma busca exaustiva de parâmetros, a fim de extrair um maior desempenho. Foi observado que o ganho se mostrou mínimo após a otimização, em parte porque os melhores parâmetros já se encontravam como os valores padrões para a primeira rodada de treinamento.

Por fim, o sistema de recomendação baseado em filtro colaborativo foi aplicado a um novo usuário que não estava presente no conjunto inicial de dados. Para tal, fez-se necessário o re-treino do modelo, que se mostrou extremamente rápido e eficiente nesta etapa (5).

Embora ainda mais dados contextuais estivessem disponíveis nos conjuntos de dados sobre os animes e sobre os usuários (seção 4.1, conjuntos 1 e 2), estes não foram utilizados no trabalho, visto que extrapolariam o contexto de filtro colaborativo. Tal limitação resultou em um modelo que obteve um F1-Score de 0.895683 e um RMSE de *1.227847*, valor que se comparados com o resultado de trabalhos similares aplicados a outros conjuntos de dados, mostram que ainda existem margens para melhorias de desempenho.

Para trabalhos futuros, é possível desenvolver sistemas de recomendação mais robustos se utilizando de estratégias híbridas, que sejam capazes de incrementar o vasto conhecimento de feedbacks implícitos contido nos demais conjuntos de dados. Ademais, um melhor ajuste dos feedbacks explícitos pode contribuir para um melhor mapeamento das interações.

REFERÊNCIAS

[1] Rosa Escandon. The film industry made a record-breaking $100 billion last year. Disponível em: <https://www.forbes.com/sites/rosaescandon/2020/03/12/the-film-industry-made-a-record-breaking-100-billion-last-year>. Acesso em: 2024-07-26.

[2] Gabriel Avila. Crunchyroll e funimation se tornam uma só empresa após compra bilionária. Disponível em: <https://jovemnerd.com.br/noticias/animes-e-mangas/crunchyroll-e-funimation-se-tornam-uma-so-empresa-apos-compra-bilionaria>. Acesso em: 2024-07-27.

[3] Sam Nussey. Sony vê impulso ao crescimento com serviço de streaming crunchyroll por globalização do anime. Disponível em: <https://www.terra.com.br/economia/sony-ve-impulso-ao-crescimento-com-servico-de-streaming-crunchyroll-por-globalizacao-do-anime,915460aa1f5006cc71eb28d669e9d5cfke9g70z3.html>. Acesso em: 2024-07-27.

[4] Francesco Ricci, Lior Rokach, and Bracha Shapira, editors. *Recommender Systems Handbook*. Springer, New York, NY, 2 edition, November 2015.

[5] Joyce van der Voet. How netflix is changing the entertainment industry. Disponível em: <https://ebfgroningen.nl/ebf-journal/how-netflix-is-changing-the-entertainment-industry>. Acesso em: 2024-07-25.

[6] Xiaofeng Yuan, Lixin Han, Subin Qian, Licai Zhu, Jun Zhu, and Hong Yan. Preliminary data-based matrix factorization approach for recommendation. *Information Processing and Management*, 58(1), January 2021.

[7] Reynaldi and Wirawan Istiono. Content-based filtering and web scraping in website for recommended anime. *Asian Journal of Research in Computer Science*, 15(2):32–42, Mar. 2023.

[8] Badal Soni, Debangan Thakuria, Nilutpal Nath, Navarun Das, and Bhaskarananda Boro. Rikonet: A novel anime recommendation engine. *Multimedia Tools and Applications*, pages 1–20, 2021.

[9] Christopher Gavra Reswara, Josua Nicolas, Mario Ananta, and Felix Indra Kurniadi. Anime recommendation system using bert and cosine similarity. In *2023 4th International Conference on Artificial Intelligence and Data Sciences (AiDAS)*, pages 109–113, 2023.

[10] MyAnimeList Co. Ltd. ©2024 All Rights Reserved. Disponível em: <https://myanimelist.net>. Acesso em: 2024-07-24.

[11] Michael J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artif. Intell. Rev.*, 13(5–6):393–408, dec 1999.

[12] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009, jan 2009.

[13] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, dec 1992.

[14] Robin Burke. *Hybrid web recommender systems*, page 377–408. Springer-Verlag, Berlin, Heidelberg, 2007.

[15] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12:331–370, 2002.

[16] Mark Claypool, Anuja Gokhale, Tim Miranda, Pavel Murnikov, Dmitry Netes, and Matthew Sartin. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR workshop on recommender systems*, volume 60, pages 1853–1870. Citeseer, 1999.

[17] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, mar 1997.

[18] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. *Collaborative Filtering Recommender Systems*, pages 291–324. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[19] Diane Kelly and Jaime Teevan. Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum*, 37(2):18–28, sep 2003.

[20] Douglas W Oard, Jinmook Kim, et al. Implicit feedback for recommender systems. In *Proceedings of the AAAI workshop on recommender systems*, volume 83, pages 81–83. Madison, WI, 1998.

[21] Benjamin Marlin, Richard S. Zemel, Sam Roweis, and Malcolm Slaney. Collaborative filtering and the missing at random assumption, 2012.

[22] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. *Proceedings of KDD Cup and Workshop*, 01 2007.

[23] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, page 426–434, New York, NY, USA, 2008. Association for Computing Machinery.

[24] Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, ICDM '05, page 625–628, USA, 2005. IEEE Computer Society.

[25] Arindam Banerjee, Inderjit Dhillon, Joydeep Ghosh, Srujana Merugu, and Dharmendra S. Modha. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, page 509–514, New York, NY, USA, 2004. Association for Computing Machinery.

[26] Quang Thang Le and Minh Phuong Tu. Active learning for co-clustering based collaborative filtering. In *2010 IEEE RIVF International Conference on Computing Communication Technologies, Research, Innovation, and Vision for the Future (RIVF)*, pages 1–4, 2010.

[27] Matěj Račinský. Myanimelist dataset. Disponível em: <https://www.kaggle.com/dsv/45582>, 2018. Acesso em: 2024-07-26.

[28] Python. Disponível em: <https://www.python.org/>. Acesso em: 2024-07-27.

[29] Surprise. Disponível em: <https://surpriselib.com/>. Acesso em: 2024-07-27.

[30] Numpy. Disponível em: <https://numpy.org/>. Acesso em: 2024-07-27.

[31] Pandas. Disponível em: <https://pandas.pydata.org/>. Acesso em: 2024-07-27.

[32] SciKitLearn. Disponível em: <https://scikit-learn.org/>. Acesso em: 2024-07-27.

[33] Stephen Bates, Trevor Hastie, and Robert Tibshirani. Cross-validation: What does it estimate and how well does it do it? *Journal of the American Statistical Association*, 119(546):1434–1445, May 2021.