MÁRCIO CÉSAR GUIMARÃES DE LIMA JUNIOR
RAFAEL VARGAS MESQUITA DOS SANTOS

# IMPLEMENTAÇÃO DE ALGORITMOS DE MEDIDAS DE JUSTIÇA SOCIAL EM SISTEMAS DE RECOMENDAÇÃO

Quipá
Editora

# IMPLEMENTAÇÃO DE ALGORITMOS DE MEDIDAS DE JUSTIÇA SOCIAL EM SISTEMAS DE RECOMENDAÇÃO

Quipá
Editora

Márcio Cesar Guimarães de Lima Junior
Rafael Vargas Mesquita dos Santos

# IMPLEMENTAÇÃO DE ALGORITMOS DE MEDIDAS DE JUSTIÇA SOCIAL EM SISTEMAS DE RECOMENDAÇÃO

1ª Edição

Quipá Editora
2024

Os autores gostariam de expressar sua profunda gratidão ao Instituto Federal do Espírito Santo (IFES) pelo aporte financeiro e apoio contínuo ao longo do desenvolvimento deste trabalho. Este suporte foi fundamental para a realização da pesquisa e os resultados alcançados.

Dados Internacionais de Catalogação na Publicação (CIP)

---

L732i Lima Junior, Márcio Cesar Guimarães de

Implementação de algoritmos de medidas de justiça social em sistemas de recomendação / Márcio Cesar Guimarães de Lima Junior e Rafael Vargas Mesquita dos Santos. – Iguatu, CE : Quipá Editora, 2024.

66 p. : il.

ISBN 978-65-5376-311-1

1. Programação de Computadores - Software. 2. Justiça social. I. Título.

CDD 005

---

Obra publicada pela Quipá Editora em março de 2024.

Quipá Editora
www.quipaeditora.com.br
@quipaeditora

# LISTAS DE FIGURAS

# LISTA DE TABELAS

# RESUMO

Sistemas de recomendação vieram para mudar o mundo, facilitando a vida das pessoas. Trata-se de uma tecnologia indispensável para diversas plataformas, como: Netflix, Youtube, Spotfy, plataformas de e-commerce, e etc. Sua função é fazer recomendações de produtos para o consumidor, agindo como um vendedor virtual. O papel crescente dos sistemas de recomendação em muitos aspectos da sociedade se apresenta como essencial, e considerar o uso de tais sistemas pode impactar o bem social. Várias modificações nos algoritmos de recomendação foram propostas para otimizar seu desempenho para situações sociais específicas com medidas relevantes.

O problema da injustiça em sistemas de recomendação onde usuários são agrupados em grupos privilegiados e desprivilegiados, a parte beneficiada representa apenas uma pequena proporção dos dados, e dispõem de uma qualidade de recomendação superior aos desprivilegiados. Portanto no presente trabalho, afim de medir a imparcialidade das recomendações aos usuários e grupos de usuários em sistemas de recomendação, foram propostas funções para medir a justiça social de sistemas recomendação através da polarização, justiça individual e justiça do grupo. Foi verificado até que ponto as classificações previstas para os itens variam (divergem) entre os usuários para encontrar e medida da polarização.

Essas funções consistem e estendem propostas que incluem medir a justiça individual, e justiça do grupo, buscando medir o erro das recomendações para cada indivíduo, ou erro dos grupos de pessoas. Foram também avaliados e comparados diferentes resultados das medidas entre cinco tipos de sistemas de recomendação diferentes, para assim dizer qual sistema foi menos polarizado, qual tratou melhor da justiça individual, e qual foi mais justo com um grupo de usuários.

Palavras-chave: Sistemas de Recomendação, WebService RESTful, Injustiça, Justiça Social, Justiça Individual, Justiça do Grupo, Polarização.

# SUMÁRIO

# CAPÍTULO 1

# INTRODUÇÃO

O universo digital oferece de forma abundante conteúdos como, vídeos, músicas, séries e filmes disponíveis em plataformas de stream e redes sociais, nas quais são quase infinitas as possibilidades de escolhas para os usuários. São diversos sites e perfis comerciais que oferecem seus produtos e serviços para todos os gostos e necessidades. No entanto, os usuários não costumam ter tempo suficiente para pesquisar detalhadamente cada produto ou serviço. É nesse contexto que o sistema de recomendação desempenha um papel importante. Ele pode auxiliar aos usuários a encontrar produtos e conteúdos de forma mais alinhada com seu perfil.

A lógica do sistema de recomendação se baseia em pegadas digitais deixadas pelos usuários ou avaliações de outros consumidores a respeito do produto procurado. Sistemas de recomendação podem ser definidos como sistemas que procuram auxiliar indivíduos a identificarem conteúdos de interesse em um conjunto de opções que poderiam caracterizar uma sobrecarga. São sistemas que procuram facilitar a penosa atividade de busca por conteúdo interessante (CAZELLA; REATEGUI, 2005).

Cazella, Nunes, e Reategui (2010), também destacam em outra publicação, a formação de perfis de usuários. Neste contexto, para que seja possível recomendar produtos, serviços ou pessoas a um usuário é necessário ter-se conhecimento sobre quem é este usuário. Antes mesmo de pensar em capturar e armazenar suas informações pessoais e comportamentais é necessário identificar qual o tipo de informação será relevante para a geração da recomendação visando uma eficiente personalização dos produtos, serviços e pessoas. Para a correta geração da recomendação a definição do perfil do usuário e coleta de informações é imprescindível .

Visto que de algum modo um sistema afeta a vida das pessoas, carece que seu comportamento seja justo. Ser justo é tratar os usuários de maneira semelhante,

independente do contexto de recomendação, sem favorecer ou diferenciar ninguém. É notável que o mundo é injusto e que a busca por mais justiça será infinita. Sempre haverá interesses buscando favorecimento e discriminação. Em tal mundo, injusto, onde já existem algoritmos para tomar decisões sobre a vida das pessoas, o primeiro passo para uma futura melhoria desses algoritmos, seria medir a sua justiça, tanto de forma individual, como de um grupo de indivíduos.

Sistemas de recomendação tradicionalmente recomendam itens para usuários indivi- duais. Em alguns cenários, entretanto, a recomendação para um grupo de indivíduos é mais adequada. Não existe ainda um bom volume de trabalhos científicos voltados para os chamados sistemas de recomendação para grupos. Um dos grandes desafios desses sistemas é, por exemplo, como lidar adequadamente com as preferências de cada integrante do grupo para geração de recomendação (CARVALHO; MACEDO, 2014). Já os sistemas de recomendação de filtragem colaborativa dependem de dados fornecidos pelo usuário para aprender modelos que são usados na finalidade de prever preferências desconhecidas do usuário. Como resultado, as recomendações feitas por tais sistemas podem carregar propriedades indesejáveis que são inerentes aos dados observados. Uma abordagem natural, então, é considerar as transformações de dados de entrada que melhoram essas propriedades (RASTEGARPANAH; GUM- MADI; CROVELLA, 2019). A justiça individual requer que cada indivíduo semelhante seja tratado de maneira semelhante, enquanto a justiça do grupo exige que todos os possíveis grupos de usuários existentes no contexto estudado, sejam tratados de forma semelhante. (LI et al., 2021).

Rastegarpanah, Gummadi, e Crovella (2019), definem que polarização é o grau em que opiniões, pontos de vista e sentimentos divergem dentro de uma população. Sistemas de recomendação podem capturar esse efeito por meio das avaliações que os usuários apresentam para itens. Então, em um algoritmo que tem alto grau de polarização, acaba-se criando uma "bolha". Por exemplo, um usuário que gosta de vôlei, receberia somente recomendações de vôlei. Portanto, este tipo de comportamento, inviabilizaria a recomendação de possíveis conteúdos interessantes sobre futebol ao usuário em questão.

A justiça de um sistema é um tema de crescente interesse em aprendizado de máquina,

onde um sistema de recomendação será considerado justo caso ele forneça igualdade de serviço (ou seja, precisão de previsão) para todos os usuários ou grupos de usuários.

O trabalho, portanto, baseia-se na implementação de algoritmos de medidas de justiça social, a saber: polarização, justiça individual, e justiça do grupo. Estas medidas poderão ser aplicadas em diferentes abordagens de sistemas de recomendação. Estas implementações serão disponibilizadas em um web service, visando facilitar suas utilizações por diferentes estratégias de recomendações. À partir das medidas, a justiça entre diferentes estratégias poderão ser comparadas.

## OBJETIVOS

### OBJETIVO GERAL

Elaborar um algoritmo para medir justiça individual, justiça do grupo e polarização das recomendações geradas por diferentes sistemas de recomendação.

### Objetivos Específicos

O presente trabalho considera os seguintes itens em seus objetivos específicos:

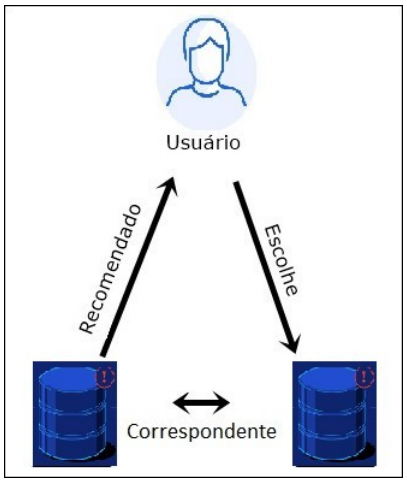
- Elaborar algoritmos de medidas de justiça social: polarização, justiça individual, e justiça do grupo;
- Incluir os algoritmos em um web sevice de recomendação;
- Aplicá-los em diferentes contextos de sistemas de recomendação: filtragem colabo- rativa, filtragem baseado em conteúdo, híbrido ponderado, híbrido misto, e híbrido de combinação sequencial;
- Medir e comparar os índices das medidas de justiça social resultantes desses diferentes tipos de sistemas de recomendação;

## SISTEMAS DE RECOMENDAÇÃO

Sistemas de recomendação são definidos como uma estratégia de tomadas de decisões para usuários em ambientes de informações complexas. Em uma época na qual a quantidade e disponibilidade de informações vem sendo cada vez maior, o processo de tomada de decisão muitas vezes, acaba tornando-se mais complexo para o usuário que precisa lidar com essa volumosa quantidade de informação.

A utilização de sistemas de recomendação permite que o usuário consiga lidar com grandes quantidades de informação, provendo recomendações personalizadas e exclu- sivas do conteúdo analisado (ISINKAYE; FOLAJIMI; OJOKOH, 2015).

Figura 1 – Exemplo de recomendação para usuário




Fonte: Autor

A maioria dos sistemas de recomendação, normalmente, aprende com as interações e preferências anteriores do usuário para recomendar itens (filmes, produtos, etc.) aos usuários. O sucesso de um algoritmo de recomendação é geralmente avaliado com a precisão de suas recomendações, ou seja, quão bem o algoritmo prevê se um usuário gostará de um item ou não - sua utilidade (LEONHARDT; ANAND; KHOSLA, 2018).

Em geral, sistemas de recomendação podem ser definidos como sistemas de

suporte que ajudam os utilizadores a encontrar informações, produtos e serviços, agregando e analisando sugestões de outros utilizadores como avaliações e atributos do utilizador (RODRIGUES, 2019).

Em um sistema típico, as pessoas fornecem avaliações de itens como entradas as quais o sistema agrega e direciona para os indivíduos considerados potenciais interessados neste tipo de recomendação. Um dos grandes desafios deste tipo de sistema é realizar a combinação adequada entre as expectativas dos usuários (seu perfil) e os itens recomendados aos mesmos, ou seja, definir e descobrir este relacionamento de interesses é o grande problema(CAZELLA; NUNES; REATEGUI, 2010).

Existem diferentes abordagens usadas no desenvolvimento de sistemas de recomendação, entre os principais estão: filtragem baseada em conteúdo[2], filtragem colaborativa[1], e filtragens híbridas[3] (ANANDHAN et al., 2018).

## FILTRAGEM COLABORATIVA

Sistemas de recomendação baseados em filtragem colaborativa fazem recomendações identificando similaridade entre usuários e itens. Assim, itens são recomen- dados aos usuários baseado na preferência de usuários semelhantes (TREICHEL, 2016). Em um sistema típico de filtragem de informação, as pessoas fornecem avaliações como entradas e o sistema agrega e direciona para os indivíduos que são considerados potenciais interessados (CAZELLA et al., 2012).

A filtragem colaborativa, por ser uma filtragem baseada na similaridade entre os usuários e itens, e não levar em consideração o conteúdo do item, é aplicável facilmente a qualquer domínio de problema (MENDES; SANTOS; PICOLI, 2018).

## FILTRAGEM BASEADA EM CONTEÚDO

A Filtragem baseada em conteúdo se constitui como sistemas que aplicam o reconhecimento de conteúdos que podem ter interesse comuns implícitos ou explícitos. Seu processo se dá por abordagens distintas, mas com finalidade principal de recomendação (PONTES et al., 2014).

Esta categoria tem sua raiz na área de recuperação e filtragem de informação, focando assim na recomendação de itens que contém informações textuais, tais como documentos e sites da Web. Conteúdos nesse tipo de filtragem são geralmente descritos por palavras-chave (BARBOSA, 2014). Um serviço de recomendação baseado em conteúdo pode ser aplicado em diferentes domínios de problema, desde que a caracterização dos itens seja adaptada a cada contexto específico (BERTOLACI, 2019).

## FILTRAGEM HÍBRIDA

A Filtragem híbrida admite a possibilidade de unir mais de uma técnica de filtragem, que oportuniza a utilização simultânea de duas ou mais. Barbosa (2014), frisa estratégias que podem ser utilizadas na recomendação híbrida, entre elas temos:

a) Ponderada: Nesta abordagem as filtragens colaborativa e baseada em conteúdo são aplicadas de forma separada, sendo que, após a geração das recomenda- ções, é realizado um processo de combinação linear, utilizando os resultados gerados (LORENÇÃO, 2021). Esta forma é considerada a mais simples da filtragem híbrida (MANSUR; PATEL; PATEL, 2017).

b) Mista: Nesta abordagem as recomendações geradas são mescladas para gera- ção do resultado final. Desde modo, o resultado apresentado ao usuário será uma lista com os dados gerados na recomendação colaborativa e baseada em conteúdo (LORENÇÃO, 2021).

c) Combinação sequencial: Nesta abordagem a filtragem baseada em

conteúdo cria os perfis dos usuários e, posteriormente, estes perfis são usados no cálculo da similaridade da filtragem colaborativa. Um recomendador refina a recomenda- ção dada por outro, utilizando a seleção recomendada como entrada novamente (BURKE, 2002). Pode-se dizer também que nesta técnica, um sistema de recomendação é influenciado pelo anterior, fazendo com que o próximo seja tendencioso ao resultado do anterior (AGGARWAL, 2016). Na filtragem híbrida por combinação sequencial, é feita a filtragem por conteúdo. Após esta pri-meira etapa, todos os perfis de usuários são gerados. Em seguida, é realizado processo de identificação de vizinhança da filtragem colaborativa (VIANA, 2022).

## MATRIZ DE AVALIAÇÃO *X*

Também conhecida como matriz original, responsável por guardar/armazenar as notas (avaliações) dos usuários, sendo estritamente necessária para o planejamento, e organização de uma avaliação. Este modelo é usado por um sistema de recomendação, servindo de entrada para que através dele, possam ser formadas predições de saída (AGGARWAL et al., 2016).

Informalmente, os problemas de recomendações podem ser reduzidos a um problema de estimar avaliações com base na matriz original para os itens que não foram ex- perimentados por um usuário. Essa estimativa é geralmente baseada na avaliação dada por este usuário para outros itens, e algumas outras informações relacionadas ao contexto (MEDEIROS, 2013).

Figura 2 - Exemplo da matriz de avaliações

| MATRIZ DE AVALIAÇÃO *X* | | | | | |
|---|---|---|---|---|---|
| ITENS | USUÁRIOS | | | | |
| | U1 | U2 | U3 | U4 | U5 |
| Evidências | | 4 | | 4 | |
| Boate Azul | 5 | | 4 | | |
| Só Hoje | | 2 | | | 5 |
| À sua maneira | 3 | | | 5 | |

Fonte: Autor

## MATRIZ ESTIMADA $\hat{X}$

Cada modelo de sistema tem sua maneira de estimar recomendações, e de estabelecer se um item é adequado para um usuário, um método simples é equiparar a média dos valores do vetor com a média dos valores do vetor do item. Existem abordagens mais complexas que usam métodos de aprendizado de máquina para estimar a probabilidade de o usuário gostar do item. Este método simples tem a desvantagem de ser necessário a coleta de dados sobre o usuário para a construção de seus perfis. Além disso, esse método depende do item, ou seja, dispõe de mais eficiência na recomendação de itens semelhantes aos já avaliados pelo usuário em sua matriz original.

Figura 3 – Exemplo da matriz estimada

| MATRIZ ESTIMADA $\hat{X}$ | | | | | |
|---|---|---|---|---|---|
| ITENS | USUÁRIOS | | | | |
| | U1 | U2 | U3 | U4 | U5 |
| Evidências | 4 | 4 | 4 | 4 | 4 |
| Boate Azul | 5 | 5 | 4 | 4,1 | 3,55 |
| Só Hoje | 1,4 | 2 | 1,3 | 5 | 5 |
| À sua maneira | 3 | 2,66 | 1,7 | 5 | 4,1 |

| LEGENDA | |
|---|---|
| | Células de Avaliações conhecidas |
| | Células de Avaliações não conhecidas |

Fonte: Autor

De modo geral a matriz estimada é a matriz de saída de um sistema de recomendação, sendo normalmente, uma lista de itens recomendados, ordenada pelo grau estimado de preferência do usuário. Ela é composta por células de avaliações conhecidas (avaliadas por usuários em sua matriz original) e não conhecidas (notas que o sistema estimou com base nas já avaliadas). Uma vez que as avaliações para os itens ainda não avaliados podem ser calculadas, os itens de maiores avaliações estimadas podem ser recomendados para o usuário.

## MATRIZ DE AVALIAÇÃO DAS RECOMENDAÇÕES $\tilde{X}$

A matriz de avaliação das recomendações é usada caso o sistema de recomendação não

seja baseado em modelo, e sim em memória. Tal que, após preenchidas as avaliações da matriz original ($X$ ), são criados “buracos” nesta matriz, para que as recomendações inteirem esses espaços vazios. E esses dados que foram retirados da matriz de avaliação ($X$ ), ficam guardados na matriz de recomendação de avaliações ($\tilde{X}$ ) para futuras comparações do que se foi recomendado, com o que foi anteriormente avaliado.

Figura 4 – Matriz de avaliação das recomendações

| MATRIZ DE AVALIAÇÃO DAS RECOMENDAÇÕES $\tilde{X}$ | | | | | |
|---|---|---|---|---|---|
| ITENS | USER | | | | |
| | U1 | U2 | U3 | U4 | U5 |
| Evidências | 2 | | 5 | | 3 |
| Boate Azul | | 4 | | 2 | 3 |
| Só Hoje | 1 | | 1 | 3 | |
| À sua maneira | | 1 | 1 | | 3 |

## MEDIDAS DE JUSTIÇA

À medida que os usuários dependem cada vez mais dos sistemas de recomendação para fazerem escolhas que afetam a vida, preocupações estão sendo levantadas sobre seu potencial inadvertido de dano social. Recentemente, estudos mostraram como os sistemas de recomendação, prevendo as preferências do usuário, podem oferecer qualidade de serviço injusta ou desigual para indivíduos (ou grupos de usuários) (BURKE; SONBOLI; ORDONEZ-GAUGER, 2018), ou levarem à polarização social, aumentando a divergência entre as preferências do indivíduo (ou grupos de usuários) (DANDEKAR; GOEL; LEE, 2013).

O sucesso de um algoritmo de recomendação é geralmente avaliado com a precisão de suas recomendações, ou seja, quão bem o algoritmo prevê se um usuário gostará de um item ou não - sua utilidade. O aspecto de justiça do usuário surge quando a tarefa exige considerar os impactos díspares das recomendações em algumas classes de usuário. A justiça de um sistema de recomendação é responsável por fornecer precisão de previsão para todos os usuários ou todos os grupos de usuários (RASTEGARPANAH; GUMMADI; CROVELLA, 2019).

## INJUSTIÇA

Quando um algoritmo recomenda um item para usuários, pode acontecer de sua recomendação ficar enviesada (retorcida). Por exemplo, imagine um grupo de homens e mulheres, e o sistema fazendo recomendações de produtos. Considerando que as recomendações feitas são analisadas, suponha que as melhores recomendações foram realizadas para os homens, e consequentemente, as piores para as mulheres. Ou em outro cenário, de e-Commerce, na qual as melhores recomendações tenham sido vinculadas aos indivíduos que compram mais, e as piores aos que compram menos. Isso significa que os aspectos relacionados ao que chamamos de justiça social não estão sendo considerados pelo algoritmo. O algoritmo está simplesmente recomendando, contudo não faz um pós processamento das recomendações buscando nivelar a qualidade das mesmas a todos usuários. Consequentemente distribuições injustas nos sistemas de recomendação, se apresentam como o grande problema do aspecto de objetivo social. A distribuição de recomendações distorcidas de itens aos usuários, causa injustiça no mercado. Portanto, certos itens acabam sendo recomendados com menor qualidade para um indivíduo ou um grupo de usuários. Ou até mesmo, uma menor frequência das recomendações a estes.

O aspecto de justiça do usuário surge quando a tarefa exige considerar os impactos díspares das recomendações em algumas classes de usuário. Por um lado, pode ser injusto ignorar os desejos de uma determinada classe de usuários ao tentar melhorar a diversidade nas recomendações, por outro lado, é igualmente injusto para os usuários se houver falta de novidades nos itens que lhes são recomendados. Um exemplo terrível, poderia ser visto em um site de recomendação de emprego, onde um usuário pouco confiante, por sempre selecionar empregos com menor salário, consequente- mente sempre será recomendado empregos com menor distribuição salarial a ele, independentemente de suas qualificações. O enigma utilidade-justiça é central para todas as medidas baseadas na justiça. Especificamente, tornar as recomendações jus- tas sempre resultará em uma certa diminuição na utilidade do sistema (LEONHARDT; ANAND; KHOSLA, 2018).

Ao projetar medidas de justiça, é necessário considerar critérios de justiça que não

discriminem injustamente um determinado indivíduo ou um conjunto de usuários. Leonhardt, Anand e Khosla (2018), também destacam o tratamento diferenciado de dois usuários ou dois itens. Segundo os autores, para melhorar a diversidade agregada, uma loja online pode recomendar itens altamente avaliados para um conjunto de usuários que são compradores em potencial, digamos os usuários ricos, enquanto novos itens (cuja qualidade não pode ser julgada) são recomendados apenas para usuários pobres. Por um lado, o sistema de recomendação pode ser injusto com o conjunto de usuários pobres, por outro lado, ele introduz disparidade de itens ao recomendar novos itens apenas para usuários que podem não realmente comprá-los, como recomendações de produtos não qualificados para usuários pobres. Por exemplo, um site de venda de produtos. Usuários ricos, que compram muito, e são muito ativos no sistema, recebem recomendações mais interessantes e mais qualificadas, que usuários que compram menos, e que possuem pouca interação com o sistema (usuários de baixa renda). Ou seja, cria-se uma injustiça por conta da diferença de qualidade dessas recomendações geradas.

## EFICIÊNCIA X MEDIDAS DE JUSTIÇA

A eficiência de um algoritmo de recomendação não está necessariamente associada a medida de justiça. Por exemplo, vamos considerar uma recomendação de música realizada pelo algoritmo de recomendação do Spotfy. Caso os dois usuários ouçam a música e não gostem, significa, de forma simplificada, que o algoritmo não foi eficiente. No entanto, houve justiça, pois, o erro na recomendação foi próximo para os dois usuários. Quando refere-se medir a justiça de um sistema de recomendação, não se trata de acertar a recomendação, e sim de proximidade do erro da recomendação para um indivíduo ou para um grupo, não possuindo relação com a eficiência do algoritmo. É a forma que o a recomendação errou ou acertou na mesma proporção.

LI, et al (2021), salienta sobre a existência de duas estruturas básicas adotadas em estudos recentes sobre discriminação algorítmica: justiça individual e justiça de grupo. Resultados das medidas de justiça são baseados no princípio de que, quanto menor variabilidade, mais justo é o sistema. Isso vale para um indivíduo, ou para um grupo específico de usuários.

## VARIÂNCIA $\sigma^2$

Dado um conjunto de dados, a variância é uma medida de dispersão que mostra o quão distante cada valor desse conjunto está do valor central (médio). Quanto menor é a variância, mais próximos os valores estão da média, mas quanto maior ela é, mais os valores estão distantes da média (RIBEIRO, 2022).

Assumindo que os dados são uma população, será utilizado o cálculo da variância populacional, que é obtido através da soma dos quadrados da diferença entre cada valor e a média aritmética, dividida pela quantidade de elementos observados (SAMPAIO; ASSUMPÇÃO; FONSECA, 2018).

Figura 5 - Exemplo de variância

| VARIÂNCIA POPULACIONAL ($\sigma^2$) | | | | | | | |
|---|---|---|---|---|---|---|---|
| ITENS | USUÁRIOS | | | | | | |
| | U1 | U2 | U3 | U4 | U5 | Média | Variância |
| Guitarra | 4 | 4 | 4 | 4 | 4 | 4,0 | 0,000000 |
| Bateria | 5 | 5 | 4 | 4,1 | 3,55 | 4,3 | 0,333600 |
| Pedestal | 1,4 | 2 | 1,3 | 5 | 5 | 2,9 | 2,886400 |
| Violão | 3 | 2,66 | 1,7 | 5 | 4,1 | 3,3 | 1,317856 |

Fonte: Autor

Como demonstra a ilustração, U1, U2, U3, U4, U5, são representados por usuários. Guitarra, bateria, pedestal, e violão correspondem os itens. Então, em termos de variabilidade, baseando-se na figura 5, o sistema gerou uma recomendação igual para cinco usuários sobre o item guitarra, 4, 4, 4, 4, 4. Neste caso, a recomendação não obteve variação, sendo assim, não aumentou, nem diminuiu, manteve a mesma nota. Já no exemplo do item pedestal, foi posssível observar que o mesmo foi o item de maior variabilidade da figura.

## POLARIZAÇÃO

Polarização refere-se ao grau em que as opiniões, visões e sentimentos divergem dentro de uma população. Diversos trabalhos anteriores levantaram e exploraram as

preocupações de que os sistemas de recomendação podem aumentar a polarização so- cial, adaptando as recomendações às preferências individuais do usuário e prendendo os usuários em suas próprias “bolhas de filtro personalizadas” (RASTEGARPANAH; GUMMADI; CROVELLA, 2019). Ela está associada à divergência ou ao aumento da divergência.

A polarização é caracterizada quando um sistema tem como resultado recomendações muito díspares, por exemplo, o algoritmo está recomendando o item guitarra, em uma escala de “1 a 5”. O mesmo recomenda com nota “1” para um usuário e nota “5” para outro. Isso significa que o algoritmo está polarizado, ou seja, existe uma variância ampla entre as recomendações que o algoritmo faz no contexto daquele item em específico. Outro exemplo, um usuário que costuma ouvir músicas no Spotfy, e sua preferência musical é rock (mas não quer dizer que ele não goste de outros estilos). Então, por ele ouvir a maioria das vezes músicas de rock, este usuário não receberá nenhuma recomendação de outros estilos mesmo que, por ventura, o conteúdo possa interessá-lo. Portanto, um algoritmo que possui alto grau de polarização acaba-se criando uma bolha dentro das recomendações do usuário. Usuários que ouvem rock, só recebem recomendações de rock, e assim nunca terão acesso a um conteúdo de outros estilos musicais, por exemplo.

Para formalizar essa noção, polarização foi definida em termos da variabilidade (vari- ância) das classificações previstas quando comparadas entre os usuários. Nas quais foram notadas que tanto a variabilidade muito alta, quanto uma variabilidade muito baixa de classificações, podem ser indesejáveis. No caso de alta variabilidade, os usuários têm opiniões fortemente divergentes, levando a conflito (RASTEGARPANAH; GUMMADI; CROVELLA, 2019).

## JUSTIÇA INDIVIDUAL

A justiça pode ser individual, visando cada usuário individualmente. Ela requer que cada indivíduo semelhante seja tratado de maneira semelhante, o que é difícil de definir precisamente devido à falta de acordo sobre as métricas de semelhança específicas de tarefas para indivíduos (LI et al., 2021).

Tomando o exemplo do Spotfy, na seção 3.2.2. No caso da justiça individual esse erro já se torna mais específico (o quanto foi essa margem de erro em relação a esses usuários). Por exemplo, a recomendação errou com nota 3 (em uma escala de 1 a 5), para o usuário a, e com nota 1 para o usuário b. Isso é um caso de injustiça individual pois, apesar do algoritmo errar para ambos usuários, ele errou muito para o usuário a, enquanto errou pouco para usuário b.

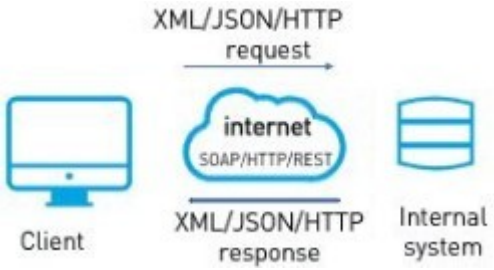
### JUSTIÇA DO GRUPO

Está relacionada a um grupo em específico, como por exemplo, homens e mulheres, pessoas que compram mais, ou pessoas que compram menos. A justiça do grupo exige que os grupos protegidos sejam tratados de forma semelhante ao grupo privilegiado ou às populações como um todo. A perspectiva de justiça do grupo para a aprendizagem supervisionada geralmente implica restrições, como probabilidades equalizadas (LI et al., 2021). Por exemplo, usuários que ouvem mais músicas no Spotfy, e possuem maior interação com o sistema, acabam recebendo melhores recomendações, que pessoas que ouvem poucas músicas, por serem menos ativas nos sistemas. Ou seja, acaba-se criando uma injustiça do grupo por conta da diferença de qualidade das recomendações geradas para o grupo menos privilegiado.

### WEB SERVICES

Web services podem ser vistos como softwares fornecidos através da rede (como a internet, por exemplo). São entidades executáveis que funcionam de maneira mo- dular e independente, sendo acessadas e publicadas em uma rede (FALTER et al., 2009). Feito para suportar interações entre máquinas de forma interoperável pela rede, podendo ser utilizadas via Web APIs (Application Programming Interface - Interface de Programação de Aplicações), se comunicando via linguagens de marcação em texto como JSON (JavaScript Object Notation) ou XML (eXtensible Markup Language) (FIELDING; RESCHKE, 2014).

Figura 6 – Exemplo de arquitetura web service




Fonte: (JUNIOR; SANTOS, 2019)

Representation State Transfer (REST), ou, se traduzirmos para português, Transferên- cia de Estado representacional. É um estilo de arquitetura de software cada vez mais utilizado no mundo inteiro, principalmente para criar serviços web e auxiliar na inte- gração de sistemas. O termo RESTful foi criado para indicar que determinado sistema ou sistema segue os princípios do REST. Por exemplo, GET, POST, PUT, DELETE (LECHETA, 2015).

Tabela 1 – Métodos Rest

| **Método** | **Descrição** |
|---|---|
| GET | Utilizado para consulta |
| POST | Utilizado para inserir |
| PUT | Utilizado para atualizar |
| DELETE | Utilizado para deletar |

Fonte: (LECHETA, 2015)

Em seu livro "Web Services RESTful: aprenda a criar web services RESTful em Java na nuvem do Google", Lecheta (2015), também conclui que, um web service é RESTful se ele tiver sido construído com base nos seguintes conceitos de REST:

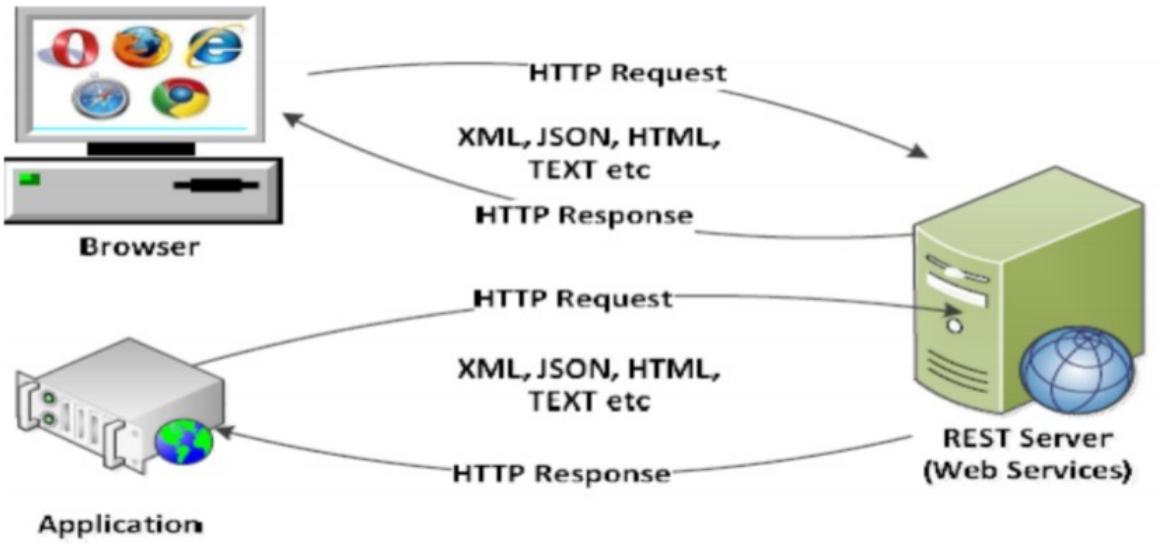
- Cada requisição ao serviço deve retornar de forma independente e não deve ter estado. Para atingir esse objetivo, podemos utilizar o protocolo HTTP, o qual já é amplamente adotado pela internet;

- Tem um conjunto de operações padronizadas pelas requisições do tipo, GET, POST, PUT, e DELETE;

- Utiliza uma URI (Uniform Resoucec Identtifier), ou se traduzirmos, uma sintaxe universa para identificar recursos. No REST, cada recurso deve conter uma URI para que seja possível consultar informações sobre ele;

- Utilização de tipos de conteúdo (mime-type) para solicitar conteúdo e retornar con- teúdo. Esse recurso é fantástico por que permite que o cliente do serviço especifique se deseja que o conteúdo seja retornado em XML ou JSON, por exemplo;

Figura 7 – Exemplo de arquitetura REST




(THU; AUNG, 2015)

# CAPÍTULO 2

# ARQUITETURA

Para elaboração dos algoritmos propostos, foi utlizado uma arquitetura de web service baseada em sistemas distribuídos, que a partir do protocolo HTTP comunica-se entre seus componentes. Nesse ambiente estão implementados 5 diferentes abordagens de sistemas de recomendação, sendo elas:

1. Sistema de recomendação com filtragem colaborativa (MENDES; SANTOS; PICOLI, 2018);

2. Sistema de recomendação com filtragem baseado em conteúdo (BERTOLACI, 2019);

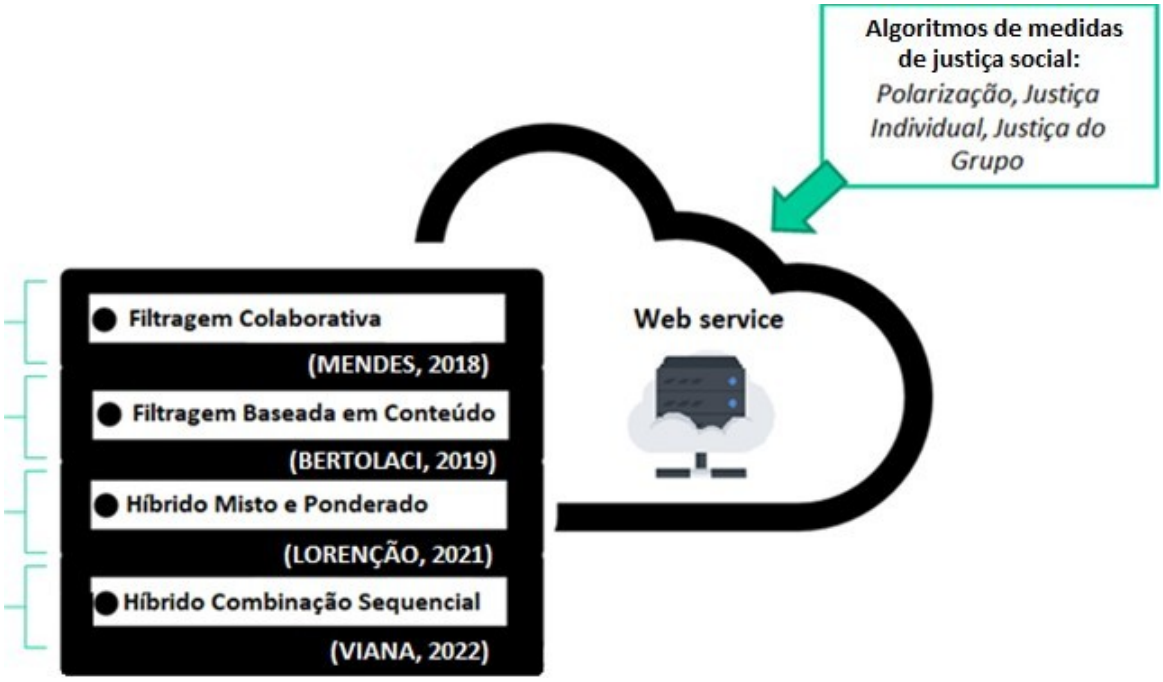
3. Sistema de recomendação com filtragem híbrida ponderada (LORENÇÃO, 2021);

4. Sistema de recomendação com filtragem híbrida mista (LORENÇÃO, 2021);

5. Sistema de recomendação híbrido baseado em combinação sequencial (VIANA, 2022);

O algoritmo de medidas de justiça será implementado no contexto do "web service de recomendação", agindo de maneira genérica no cálculo das medidas de polarização, justiça individual, e justiça do grupo. Os resultados de recomendações dos diferentes sistemas de recomendação poderão ser avaliados sob a perspectiva das medidas de justiça social, como demostrado na figura 8.

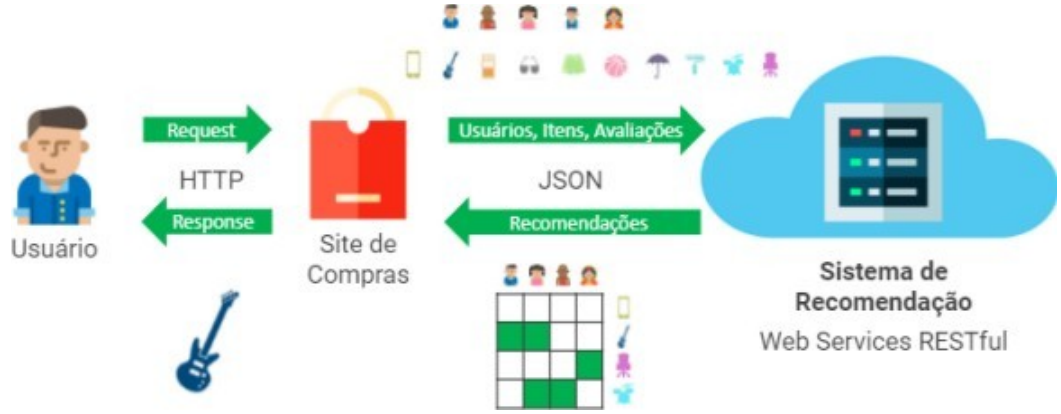
Figura 8 – Algoritmo medidas de justiça




**Exemplo de utilização do Web Service de recomendação**

Para o desenvolvimento deste projeto, alguns componentes foram planejados a partir do modelo de web services, detalhado na figura 9. Neste exemplo, a aplicação cliente é um site de compras, o qual interage com o sistema de recomendação para obter recomendações de produtos para usuários (clientes) cadastrados no site.

Figura 9 – Arquitetura do sistema




Fonte: (VIANA, 2022)

## FERRAMENTAS

Durante o decorrer do projeto, foram utilizadas as seguintes tecnologias:

- Java: Linguagem de programação multiplataforma, versão OpenJDK (18.0.2.1);

- IntelliJ IDEA 2022.2.2: Framework de desenvolvimento integrado escrito em Java para desenvolver software de computador escrito em Java, e outras linguagens baseadas em JVM;

- PostgreSQL: Banco de dados que utiliza SQL para seu funcionamento na versão 12.12;

- Postman: Software que auxilia no desenvolvimento de Api's, necessário para efetuar testes nas rotas de envio de requisições http, enviar corpo de dados e cabeçalhos de requisição, assim como é responsável por receber dados de resposta, pode ser considerado o ambiente onde se tem o design das Apis, quais dados enviar, respostas a receber, testar os endpoints, e afins (POSTMAN, 2019);

- H2: Banco de dados em memória, usado na etapa de testes;

- Git: plataforma de hospedagem de código-fonte e arquivos com controle de versão. Ele permite que programadores, utilitários ou qualquer usuário cadastrado na plataforma contribuam em projetos privados e/ou Open Source de qualquer lugar do mundo (Ferramenta para controle de versão de código);

- Máquina de testes: Os testes serão realizados em uma máquina cuja as configurações são: Windows 10 - 64 Bits, 12 GB de memória ram (DDR3-1600 Mhz), processador intel i7-3632QM CPU @2.20GHZ, ssd 240 GB sandisk plus;

## ESTRATÉGIA DE AÇÃO

Para o desenvolvimento do projeto proposto, foram realizadas as seguintes etapas:

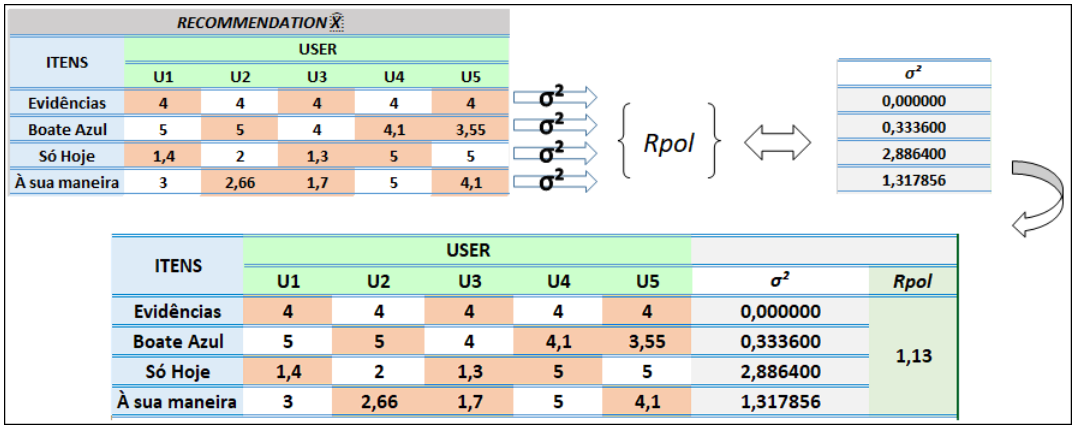
- Estudo dos principais conceitos sobre sistemas de recomendação;
- Estudo e proposição de medidas de justiça social: polarização, justiça individual e justiça do grupo;
- Implementação, utilizando a linguagem de programação java, para o cálculo das medidas de justiça social;
- Inclusão dos algoritmos em uma arquitetura de web sevice de recomendação;
- Preparação do estudo de caso 1. O dataset foi construído por meio de um experimento no contexto musical. Foram registrados cerca de 10 diferentes estilos, línguas, cantadas em solo ou banda, cantadas por homem e mulheres, etc.). Para avaliação, foram selecionados 59 avaliadores atribuindo notas em uma escala de 0 a 5;
- Utilização do dataset 1 para aplicação dos algoritmos de diferentes contextos de sistemas de recomendação: filtragem colaborativa[1], filtragem baseado em conteúdo[2], híbrido ponderado[3a], híbrido misto[3b], e híbrido combinação sequencial[3c];
- Avaliação e comparação dos índices das medidas de justiça social resultantes desses diferentes tipos de sistemas de recomendação;
- Conclusão: apresentar os resultados de medidas de justiça social geradas por diferentes tipos de sistemas de recomendação para comparação de resultados;

## ALGORITMO

No presente trabalho, foram elaborados três algoritmos para medidas de justiça social de diferentes tipos de sistemas de recomendação.

O primeiro deles é o método para medir a polarização (*Rpol*), como demonstrado na figura 10:

Figura 10 – Exemplo de funcionamento do algoritmo de polarização - *Rpol*

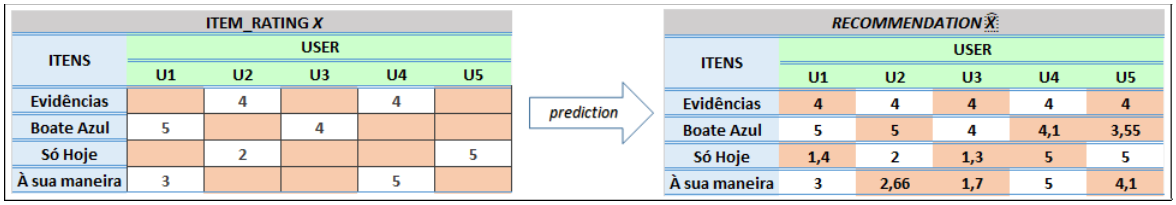



Fonte: Autor

Na primeira matriz do exemplo ($\hat{X}$), também conhecida como matriz estimada (ou matriz de recomendação), será extraída a variância populacional ($\sigma^2$) de cada item em relação a todos os usuários. No mesmo exemplo, pode-se perceber que o item 3, representado pela música "só hoje"foi estimado com notas: 1.4, 2, 1.3, 5, 5, para usuários do 1 ao 5, respectivamente, obtendo uma variância de "2.886400", tornando a música só hoje, o item mais polarizado do exemplo. Logo, através da média de todas as variâncias, foi possível encontrar e medida de polarização (*Rpol* ).

Em seguinda foi implementado o método para medir a justiça individual (*Ridnv* ). A lógica do algoritmo se baseia no exemplo da figuras 11, e 12:

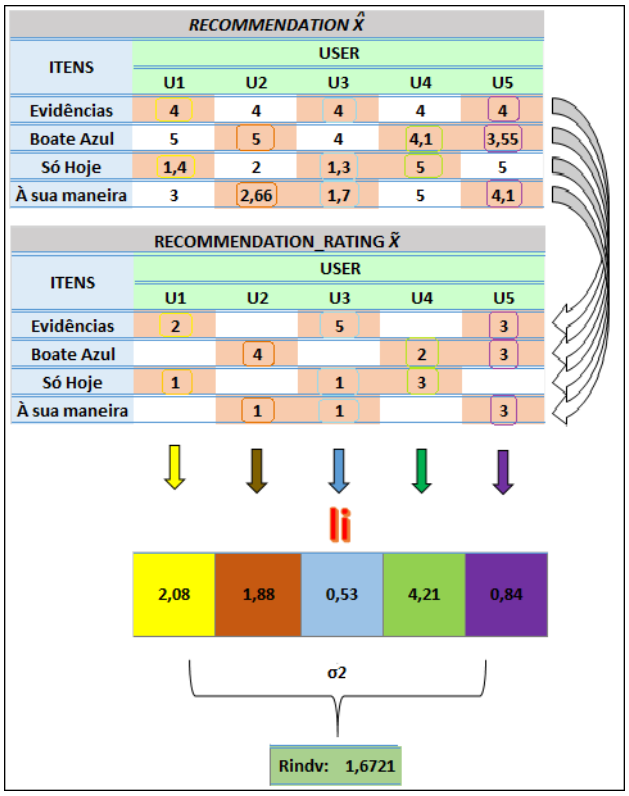
Figura 11 - Exemplo de recomendação




Fonte: Autor

A figura 11, demonstra um sistema de recomendação fazendo uma predição (uma recomendação), com base nas informações que já possui, em sua matriz original, que é denominada por "*ITEM_RATING X* " no exemplo.

Figura 12 - Exemplo de funcionamento do algoritmo de justiça individual - *Rindv*




Fonte: Autor

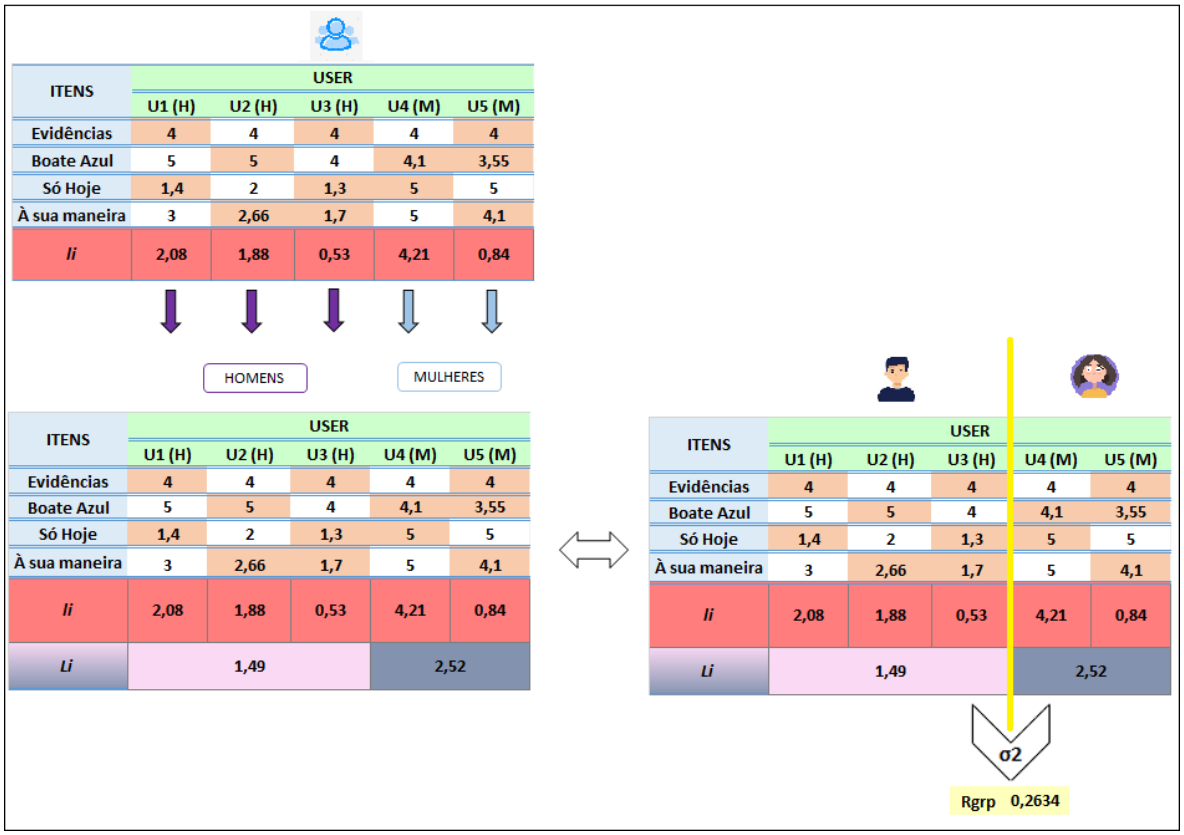
Feitas as predições, a figura 12 exemplifica como chegar ao resultado da medida de *Rindv* . Diferentemente da polarização, tal qual, é medido um item, em relação a todos os usuários, na justiça individual, é feita a comparação em relação um usuário, e seus respectivos itens.

Para o exemplo citado, estamos tratando a medida de justiça individual sendo extraída de um sistema baseado memória, e não em modelo. Portanto, será necessário a inserção de notas em uma terceira matriz (*RECOMMENDATION_RATING* $\tilde{X}$ ), suas notas são compostas por avaliações feitas por usuários, e que foram retiradas para que o sistema gerasse as recomendações nesse lugares vazios. Assim, será buscado o erro entre essas duas matrizes em relação a cada usuário.

As células com círculos amarelos da matriz "*RECOMMENDATION* $X$ ", são comparadas com os círculos amarelos da matriz "*RECOMMENDATION_RATING* $\tilde{X}$ ", em relação ao usuário 1 (4, 1.4, e 2, 1). Isto ocorre com todos os usuários e suas respectivas cores chegando ao resultado das perdas (*li* ) de cada usuário. Sendo assim, para chegar ao resultado da medida de justiça individual (*Rindv* ), é necessário extrair a variância das perdas de cada usuário.

Por fim, também foi elaborado um algoritmo de medida de justiça do grupo onde foi buscado o *Rgrp*, como demonstrado a figura 13:

Figura 13 - Exemplo de funcionamento do algoritmo de justiça do grupo - *Rgrp*




Fonte: Autor

A figura 13, primeiramente demonstra a separação dos usuários em dois grupos. Usuários 1, 2, e 3, no grupo dos homens, e usuários 4, e 5, no grupo das mulheres. Onde, diferentemente da justiça individual, tal qual, foi descoberto o erro de cada indivíduo (*Ii*), neste caso, foi buscado o erro dos grupos baseando-se na média da perda dos homens, e respectivamente na perda do grupo das mulheres. Sendo assim, para encontrar a medida de justiça do grupo (*Rgrp*), foi utilizado a variância entre os erros dos grupos (*Li* ), nesse exemplo representando por 1.49, e 2.52.

## CÁLCULO DE MEDIDAS SOCIAIS

Tendo em vista todas as especificações e discussões da seção anterior, iremos definir formalmente as métricas que especificam as funções objetivo associadas à justiça individual e justiça do grupo.

Começaremos apresentando a configuração do sistema, notação e a definição do problema. Suponha que $X \in R^{n \times d}$ seja uma matriz de classificação parcialmente obser- vada de $n$ usuários e $d$ itens, de modo que o elemento $x_{ij}$ denota a classificação dada pelo usuário $i$ para o item $j$ . Seja $\Omega$ o conjunto de índices de classificações conhecidas em $X$ . Além disso, $\Omega_i$ denota os índices de classificações de itens conhecidos para o usuário $i$ , e $\Omega_j$ denota os índices de classificações de usuários conhecidas para o item $j$ .

Dado um sistema de recomendação tradicional é gerada uma matriz estimada de recomendações $\hat{X} = [\hat{X}_{ij}]_{n \times m}$. Neste problema de recomendação supomos usuários em um conjunto $\{u_1, u_2, ..., u_n\}$ e itens em um conjunto $\{v_1, v_2, ..., v_m\}$.

## POLARIZAÇÃO

Para capturar a polarização, procuraremos medir até que ponto as avaliações do usuário discordam. Assim, para medir a polarização do usuário, consideraremos as avaliações estimadas $\boldsymbol{X}$, e $\hat{}$definiremos a métrica de polarização como a soma normalizada das

distâncias euclidianas aos pares entre avaliações estimadas do usuário, ou seja, entre as linhas de $\hat{X}$. Em particular:

$$R(\hat{X}) = \frac{1}{n^2 d} \sum_{k=1}^{n} \sum_{l>k} \| \hat{x}_k - \hat{x}_l \|^2 \quad (1)$$

O termo de normalização $\frac{1}{n^2 d}$ torna a métrica de polarização idêntica à seguinte definição:

$$R_{pol}(\hat{X}) = \frac{1}{d} \sum_{j}^{d} \sigma^2_j \quad (2)$$

onde $\sigma^2_j$ é a variação das avaliações estimadas de usuários para o item $j$. Assim, esta métrica de polarização pode ser interpretada como a média das variações das avaliações estimadas em cada item, ou equivalentemente como a discordância média do usuário sobre todos os itens.

Na figura 14 é possível observar, que a polarização é medida em relação ao item, onde será extraído a variância em relação a todas as notas daquele item em específico. Assim, acharemos o grau de polarização que a recomendação da matriz estimada obteve. De forma mais técnica, o cálculo da polarização se dá através da variância populacional entre um determinado item, e todos os usuários, considerando células de avaliações conhecidas, e não conhecidas. Logo, seu resultado (*Rpol*), é a média dos resultados obtidos de todos os itens (média de todas as variâncias).

Figura 14 - Rpol - Polarização

| POLARIZAÇÃO | | | | | | | |
|---|---|---|---|---|---|---|---|
| ITENS | USUÁRIOS | | | | | | |
| | U1 | U2 | U3 | U4 | U5 | $\sigma^2$ (Populacional) | *Rpol* |
| Evidências | 4 | 4 | 4 | 4 | 4 | 0,000000 | 1,13 |
| Boate Azul | 5 | 5 | 4 | 4,1 | 3,55 | 0,333600 | |
| Só Hoje | 1,4 | 2 | 1,3 | 5 | 5 | 2,886400 | |
| À sua maneira | 3 | 2,66 | 1,7 | 5 | 4,1 | 1,317856 | |

| LEGENDA | |
|---|---|
| $\sigma^2$ | Variância |
| ***Rpol*** | Medida de Polarização |

Fonte: Autor

## Justiça individual

Para cada usuário *i*, definimos $l_i$ , a perda de usuário *i*, como a estimativa do erro quadrático médio sobre as classificações conhecidas do usuário *i* :

$$\ell_i = \frac{||P^i(\hat{x}^i - \tilde{x}^i)||^2}{|\Omega^i|^2} \quad (3)$$

Em seguida, definiremos a injustiça individual como a variação das perdas do usuário:

$$R_{indv}(\tilde{X}, \hat{X}) = \frac{1}{n^2} \sum_{k=1}^{n} \sum_{l>k} (\ell_k - \ell_l)^2 \quad (4)$$

Para melhorar a justiça individual, buscaremos minimizar a $R_{indv}$ .

Como demonstrado na figura 15, conseguimos perceber que a justiça individual não é medida em relação ao item, e sim em relação a um determinado usuário, e todos os itens da tabela. Primeiramente busca-se o *li* , que é representado pela perda (ou erro) das comparações entre as células de avaliações conhecidas da matriz estimada, e da matriz de avaliação das recomendações.

O resultado da justiça individual (*Rindv* ), é encontrado através da variância populacional ($\sigma^2$) dos valores obtidos do ”*li* ” de cada usuário.

Figura 15 – Ridnv - Justiça individual

| JUSTIÇA INDIVIDUAL | | | | | | |
|---|---|---|---|---|---|---|
| ITENS | USUÁRIOS | | | | | |
| | U1 (H) | U2 (H) | U3 (H) | U4 (M) | U5 (M) | $\sigma^2$ |
| Evidências | 4 | 4 | 4 | 4 | 4 | 0,000000 |
| Boate Azul | 5 | 5 | 4 | 4,1 | 3,55 | 0,333600 |
| Só Hoje | 1,4 | 2 | 1,3 | 5 | 5 | 2,886400 |
| À sua maneira | 3 | 2,66 | 1,7 | 5 | 4,1 | 1,317856 |
| *li* | 2,08 | 1,88 | 0,53 | 4,21 | 0,84 | *Rindv:* 1,6721 |

| LEGENDA | |
|---|---|
| *li* | Perda (Ou erro) do usuário |
| *Rindv* | Medida de Justiça Individual |

Fonte: Autor

## JUSTIÇA DO GRUPO

Sendo $I$ o conjunto de todos os usuários / itens e $G = \{G_1, G_2, ..., G_g\}$ seja uma partição de usuários / itens em grupos, ou seja, $I = U_{i\in\{1,2,...,g\}} G_i$ . Definiremos a perda do grupo como a estimativa do erro quadrático médio sobre todas as classificações conhecidas no grupo $i$ :

$$L_i = \frac{||P_{\Omega_{G_i}}(\hat{X} - \tilde{X})||^2}{|\Omega_{G_i}|^2} \quad (5)$$

Para determinada partição $G$, a injustiça do grupo será a variação de todas as perdas do grupo:

$$R_{grp}(\tilde{X}, \hat{X}, G) = \frac{1}{g^2} \sum_{g} \sum_{k=1\ l>k} (L_k - L_l)^2 \quad (6)$$

Novamente, para melhorar a justiça do grupo, procuraremos minimizar $R_{grp}$.

Figura 16 – Rgrp - Justiça do grupo

| JUSTIÇA DO GRUPO | | | | | | | |
|---|---|---|---|---|---|---|---|
| ITENS | USUÁRIOS | | | | | | |
| | U1 (H) | U2 (H) | U3 (H) | U4 (M) | U5 (M) | $\sigma^2$ | *Rpol* |
| Evidências | 4 | 4 | 4 | 4 | 4 | 0,000000 | 1,13 |
| Boate Azul | 5 | 5 | 4 | 4,1 | 3,55 | 0,333600 | |
| Só Hoje | 1,4 | 2 | 1,3 | 5 | 5 | 2,886400 | |
| À sua maneira | 3 | 2,66 | 1,7 | 5 | 4,1 | 1,317856 | |
| *li* | 2,08 | 1,88 | 0,53 | 4,21 | 0,84 | *Rindv :* | 1,6721 |
| *Li* | 1,49 | | | 2,52 | | *Rgrp:* | 0,2634 |

| LEGENDA | |
|---|---|
| | |
| *Li* | Perda (Ou erro) do grupo usuários |
| *Rgrp* | Medida de Justiça do Grupo |

Fonte: Autor

O exemplo da figura 16, nos remete a justiça do grupo, tal qual é medida em relação ao grupo de usuários. Nesse caso, os usuários foram agrupados por sexo: H[1], e M[2], formando dois grupos. Para encontrar a medida da justiça do grupo (*Rgrp*). Primeiramente é buscado a média das perdas (*li* ) de cada indivíduo do grupo.

[1] Homens
[2] Mulheres

O cálculo da "*Li*" da justiça do grupo é obtido através da média dos resultados do "*li*" da justiça individual, de cada grupo de usuários. Ainda no exemplo da figura 16, repare que o "*li*"*s* : 2.08, 1.88, 0, 53, pertencem aos erros dos usuários 1, 2, e 3 respectivamente, por tanto, a média desses 3 valores é: 1, 49. Consequentemente o "*Li* " do grupo dos homens foi encontrado. Logo, seu resultado final (*Rgrp*) é encontrado através da variância dos valores obtidos de "*Li* " de cada grupo.

# CAPÍTULO 3

# RESULTADOS

Neste capítulo será detalhado o resultado do estudo de caso aplicado.

## API DE RECOMENDAÇÃO

Para este trabalho, foi utilizado como base o código do trabalho anterior de Vianna (2022), disponível no GitHub[1]. O código de Vianna teve de ser modificado para a inclusão das medidas de justiça, disponível também no GitHub[2]. Além disso, os endpoints da API de requisições permaneceram os mesmos do trabalho original de Vianna (2022). A documentação da API original de Vianna pode ser encontrada no Postman[3].

### Alterações no algoritmo

Não foram feitas alterações no modelo do projeto. Foram criadas novas classes controller e service para que fosse possível a captura das medidas. As mudanças aplicadas ao projeto foram as criações das classes: RpolController, RindvController, RgrpController, RpolService, RindvService, RgrpService.

As classes do tipo controller são responsáveis pelo recebimento das requisições de aplicações clientes. Já as do tipo Service, implementam de fato, o cálculo das medidas de justiça social. Todos os códigos inseridos no projeto de Vianna (2022) estão disponíveis no apêndice B.

## ESTUDO DE CASO

### Contexto cultural

Neste trabalho, foram utilizados dados originais coletados no trabalho de Lorenção (2021). Na ocasião, foi desenvolvido um website, por meio do qual usuários

avaliaram um conjunto de músicas. A tabela 2 detalha o estudo de caso original.

[1] Link do repositório original https://github.com/CaioFViana/srh-backend
[2] Link do repositório modificado: https://github.com/marciocgl/srh-backend
[3] Link dos endpoints da API: https://documenter.getpostman.com/view/16559558/2s8YzMXkQb

Tabela 2 – Dados do estudo de caso

| Itens | Quantidade |
|---|---|
| Número de avaliadores | 59 |
| Número de músicas | 10 |
| Número de avaliações | 590 |

Fonte: (LORENÇÃO, 2021)

Para os testes e avaliações dos algoritmos de medidas de justiça social, foram utilizados estes dados do estudo de caso citado anteriormente. Inicialmente os usuários realizaram avaliações para todos os itens de músicas disponíveis. Todavia, para que as recomendações fossem geradas, via algoritmos de recomendações, foram descartadas algumas destas avaliações. Para ser mais específico, neste estudo de caso, foram mantidas 8 avaliações de cada usuário, deixando 2 itens de músicas sem avaliações.

## RESULTADOS DAS MEDIDAS DE JUSTIÇA SOCIAL DOS ALGORITMOS

Neste tópico será abortado como foram realizados os cálculos das medidas de justiça social. Para o cálculo da polarização, foram utilizadas as classes RpolController e RpolService, já a medida de justiça individual, foi por meio das classes RindvController e RindvService, e as classes RgrpController e RgrpService, foram as responsáveis pela medida de justiça do grupo.

Para os resultados das medidas de polarização (*Rpol* ), primeiramente, foi gerada uma lista de itens com avaliações já estabelecidas no estudo de caso citado na tabela 2. Essas avaliações estavam guardadas no sistema na classe chamada "ItemRating". Em seguida, foi criado outra lista para capturar as recomendações feitas para cada item. Em posse das notas avaliadas e recomendadas, foi elaborada uma terceira lista para receber todas as avaliações das duas listas. Para encontrar a polarização, primeiramente foi procurado a variância dessa lista que contém as avaliações dos usuários, e as notas das

recomendações, ou seja, como são 10 itens, foram obtidas 10 variâncias. O resultado da polarização vem da média de todas as variâncias obtidas, e todo esse processo foi feito para cada um dos 5 algoritmos.

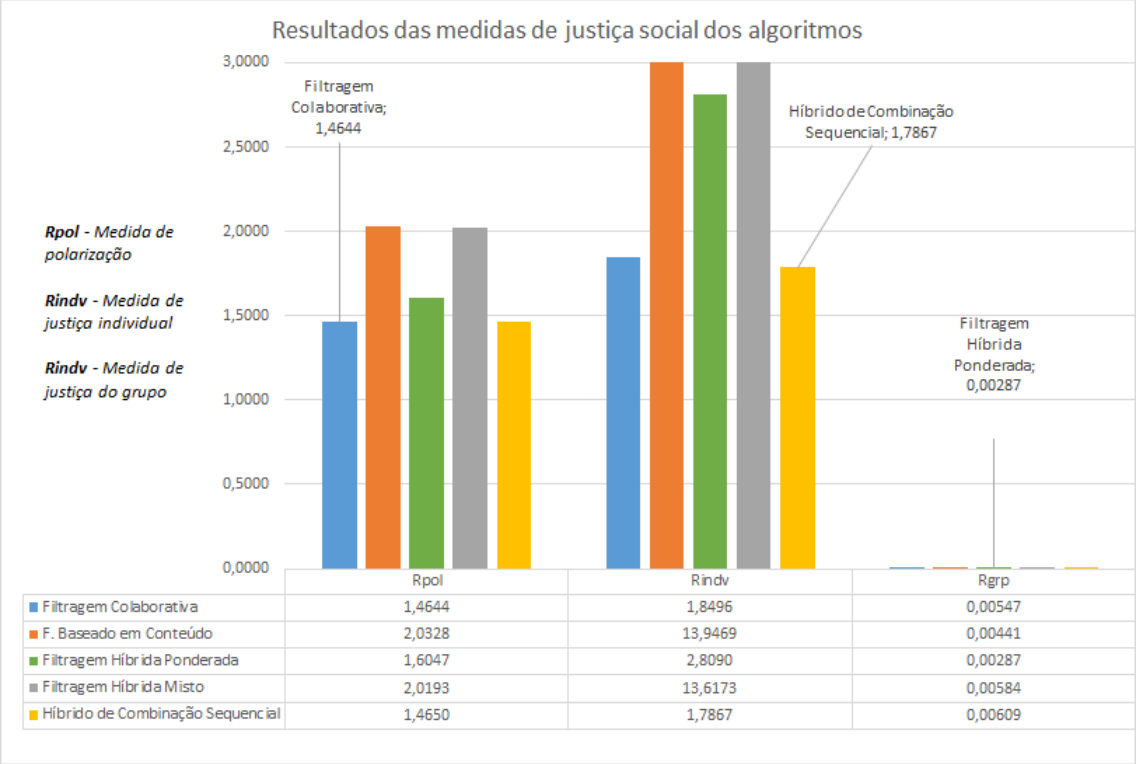
Já para obter os resultados das medidas de justiça individual (*Rindv* ), primeiramente foram geradas as recomendações da classe “Recommendation”, tal que foram feitas duas recomendações para cada um dos 59 avaliadores (Evaluator), em diferentes itens. Para busca do “*li* ” de cada avaliador, foram inseridos no H2, na classe "RecommendationRating", as avaliações que foram retiradas para que o sistema de recomendação pudesse preenche-las. Os códigos das inserções podem ser visualizados no arquivo "srh-backend" do projeto, disponibilizado no GitHub citado no tópico 5.1, todas as inserções foram baseadas no estudo de caso no contexto cultural na seção 5.2.1.

Por fim, para encontrar o “*li*” individual, foi descoberto o erro entre o que o sistema recomendou para o usuário, com o que ele tinha avaliado em seus respectivos itens (a diferença entre as classes Recommendation e RecommendationRating). E para chegar a medida de justiça individual do algoritmo (*Rindv* ), foi extraído a variância de todas essas diferenças ($li's$). Todo esse processo foi repetido para cada algoritmo.

Para os resultados das medidas de justiça do grupo (*Rgrp*), primeiramente os usuários foram classificados em dois grupos, (usuários com id ímpar, e usuários com id par). Em seguida, foi buscado a média dos ”$li's$” dos usuários que pertencem ao grupo dos impares, e depois a média dos ”$li's$” dos usuários dos grupos pares, encontrando assim, o ”*Li*” de cada grupo (a perda dos grupos). Então, para encontrar a justiça do grupo (*Rgrp*) do algoritmo, foi extraída a variância dos grupos (variância dos "$Li's$"). E novamente, todo o processo foi feito para cada algoritmo.

Em suma, a figura 17 ilustra um gráfico com os resultados encontrados de cada uma das medidas de justiça social para cada uma das 5 diferentes abordagens de sistemas de recomendação (4.1). Estas medidas estão presentes no anexo B onde está disponibilizado um link, tal qual, através dele é possível visualizar o passo a passo com maiores detalhes.

Figura 17 – Resultados das medidas de justiça social




| | Rpol | Rindv | Rgrp |
|---|---|---|---|
| Filtragem Colaborativa | 1,4644 | 1,8496 | 0,00547 |
| F. Baseado em Conteúdo | 2,0328 | 13,9469 | 0,00441 |
| Filtragem Híbrida Ponderada | 1,6047 | 2,8090 | 0,00287 |
| Filtragem Híbrida Misto | 2,0193 | 13,6173 | 0,00584 |
| Híbrido de Combinação Sequencial | 1,4650 | 1,7867 | 0,00609 |

Fonte: Autor

O menor valor de polarização (*Rpol*) foi calculado considerando as recomendações geradas pela estratégia de Filtragem Colaborativa. Isso significa que os valores de recomendações para diferentes usuários, considerando um mesmo item musical, nesta estratégia de recomendação, foram os que possuíram menor disparidade. Já o menor valor da justiça individual (*Rindv* ) foi representado pelo algoritmo Híbrido de Combina- ção Sequencial, considerando um indivíduo em relação aos itens do contexto musical. Enfim, considerando grupos pares e grupos ímpares, o sistema com abordagem Híbrida Ponderada, foi o que melhor tratou da justiça do grupo (*Rgrp*) neste contexto abordado.

# CONCLUSÃO

## ANÁLISE GERAL DO TRABALHO

O trabalho realizado gerou como artefato o aprimoramento do sistema de recomenda- ção feito por (VIANA, 2022), que, por sua vez, já era uma continuação dos estudos de (LORENÇÃO, 2021), (BERTOLACI, 2019), e (MENDES; SANTOS; PICOLI, 2018).

Sendo código aberto, este trabalho também serve o propósito de uma plataforma para estudos e melhorias que qualquer pessoa possa realizar, além de ser um código modularizado, o que permite realizar as modificações e adições.

Sobre o estudo de caso, vale ressaltar que após as análises dos resultados das medidas de justiça social dos 5 diferentes algoritmos de recomendação, passou-se a ser permitido observar que, partindo do princípio de quanto menores são os erros das avaliações, mais justo é o sistema. Foi possível concluir que o sistema com filtragem colaborativa é o sistema menos polarizado em relação aos itens, enquanto o sistema híbrido de combinação sequencial é o sistema mais justo com o usuário, ao mesmo tempo que a justiça do grupo, apesar das notas serem muito próximas, o sistema com filtragem híbrida ponderada foi o sistema que melhor tratou dessa questão.

## TRABALHOS FUTUROS

Como trabalhos futuros, pode-se considerar uma nova filtragem a ser incluída no web service descrito neste trabalho. Além disso, pode-se também considerar maneiras de encontrar melhorias nas medidas de polarização, na justiça individual, e na justiça do grupo nos sistemas de recomendação presentes no web service utilizado.

## REFERÊNCIAS


AGGARWAL, C. C. Ensemble-based and hybrid recommender systems. In: *Recommender systems*. [S.l.]: Springer, 2016. p. 199–224.

AGGARWAL, C. C. et al. *Recommender systems*. [S.l.]: Springer, 2016. v. 1.

ANANDHAN, A. et al. Social media recommender systems: review and open research issues. *IEEE Access*, IEEE, v. 6, p. 15608–15628, 2018.

BARBOSA, C. E. M. Estudo de técnicas de filtragem híbrida em sistemas de recomendação de produtos. *Monografia. Centro de Informática, Ciência da Computação, UFPE*, 2014.

BERTOLACI, L. M. Elaboração de um sistema de recomendação baseado em conteúdo. Cachoeiro de Itapemirim, 2019.

BURKE, R. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, Springer, v. 12, n. 4, p. 331–370, 2002.

BURKE, R.; SONBOLI, N.; ORDONEZ-GAUGER, A. Balanced neighborhoods for multi-sided fairness in recommendation. In: PMLR. *Conference on fairness, accountability and transparency*. [S.l.], 2018. p. 202–214.

CARVALHO, L.; MACEDO, H. Introdução aos sistemas de recomendação para grupos. *Revista de Informática Teórica e Aplicada*, v. 21, n. 1, p. 77–109, 2014.

CAZELLA, S. C. et al. Desenvolvendo um sistema de recomendação de objetos de aprendizagem baseado em competências para a educação: relato de experiências. In: *Brazilian symposium on computers in education (simpósio brasileiro de informática na educação-sbie)*. [S.l.: s.n.], 2012. v. 23, n. 1.

CAZELLA, S. C.; NUNES, M.; REATEGUI, E. A ciência da opinião: Estado da arte em sistemas de recomendação. *André Ponce de Leon F. de Carvalho; Tomasz Kowaltowski.. (Org.). Jornada de Atualização de Informática-JAI*, p. 161–216, 2010.

CAZELLA, S. C.; REATEGUI, E. B. Sistemas de recomendação. In: *XXV Congresso da Sociedade Brasileira de Computação*. [S.l.: s.n.], 2005.

DANDEKAR, P.; GOEL, A.; LEE, D. T. Biased assimilation, homophily, and the dynamics of polarization. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 110, n. 15, p. 5791–5796, 2013.

FALTER, T. et al. *System and method for a Web service definition*. [S.l.]: Google Patents, 2009. US Patent 7,620,934.

FIELDING, R.; RESCHKE, J. *Hypertext transfer protocol (HTTP/1.1): Semantics and content*. [S.l.], 2014.

ISINKAYE, F. O.; FOLAJIMI, Y. O.; OJOKOH, B. A. Recommendation systems: Principles, methods and evaluation. *Egyptian informatics journal*, Elsevier, v. 16, n. 3, p. 261–273, 2015.

JUNIOR, A. R. M. S.; SANTOS, M. dos. Arquitetura rest api e desenvolvimento de uma aplicação web service. *PROJETOS E RELATÓRIOS DE ESTÁGIOS*, v. 1, n. 1, p. 1–59, 2019.

LECHETA, R. R. *Web Services RESTful: aprenda a criar web services RESTful em Java na nuvem do Google*. [S.l.]: Novatec Editora, 2015.

LEONHARDT, J.; ANAND, A.; KHOSLA, M. User fairness in recommender systems. In: *Companion Proceedings of the The Web Conference 2018*. [S.l.: s.n.], 2018. p. 101–102.

LI, Y. et al. User-oriented fairness in recommendation. In: *Proceedings of the Web Conference 2021*. [S.l.: s.n.], 2021. p. 624–632.

LORENÇÃO, H. d. S. Elaboração de um serviço de recomendação híbrido implantado em webservice restful. Cachoeiro de Itapemirim, 2021.

MANSUR, F.; PATEL, V.; PATEL, M. A review on recommender systems. In: IEEE. *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*. [S.l.], 2017. p. 1–6.

MEDEIROS, I. Estudo sobre sistemas de recomendação colaborativos. *Acedido em março*, v. 2, p. 2020, 2013.

MENDES, T. M.; SANTOS, R. V. M. dos; PICOLI, J. G. Algoritmo de recomendação colaborativa aplicado ao contexto educacional. *Brazilian Applied Science Review*, v. 3, n. 1, p. 703–711, 2018.

PONTES, W. L. et al. Filtragens de recomendação de objetos de aprendizagem: uma revisão sistemática do cbie. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. [S.l.: s.n.], 2014. v. 25, n. 1, p. 549.

POSTMAN, I. *Postman*. 2019.

RASTEGARPANAH, B.; GUMMADI, K. P.; CROVELLA, M. Fighting fire with fire: Using antidote data to improve polarization and fairness of recommender systems. In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. [S.l.: s.n.], 2019. p. 231–239.RIBEIRO, A. G. Medidas de dispersão: variância e desvio padrão. *Brasil Escola. Disponível em< http://www. brasilescola. com/matematica/medidas-dispersao-variancia- desvio-padrao. htm>. Acesso em*, v. 17, 2022.

RODRIGUES, F. R. Sistemas de recomendação para percursos culturais. 2019. SAMPAIO, N. A. de S.; ASSUMPÇÃO, A. R. P. de; FONSECA, B. B. da. Estatística descritiva. 2018.

THU, E. E.; AUNG, T. N. Developing mobile application framework by using restful web service with json parser. In: SPRINGER. *International Conference on Genetic and Evolutionary Computing*. [S.l.], 2015. p. 177–184.

TREICHEL, T. Benchmarking e avaliação de usuários de um sistema de recomendação baseado na similaridade entre itens. 2016.

VIANA, C. F. Elaboração de um sistema de recomendação híbrido baseado em combinação sequencial. Cachoeiro de Itapemirim, 2022.

# APÊNDICE A

## CALCULANDO A PERDA DA JUSTIÇA INDIVIDUAL

Neste apêndice, será demonstrado o processo para se calcular a perda ou erro (*li* ).

O cálculo do "*li* "da justiça individual é realizado através da matriz estimada, e da matriz original (matriz de avaliações). Ele reconhece a diferença entre as células de cada matriz, sem considerar as células de avaliações não conhecidas, tal como:

| X - Original | | |
|---|---|---|
| | A | B |
| 1 | 1 | 4 |
| 2 | 2 | 5 |
| 3 | 3 | 6 |

x

| X - Estimada | | |
|---|---|---|
| | A | B |
| 1 | 7 | 10 |
| 2 | 8 | 11 |
| 3 | 9 | 12 |

Fonte: Autor

Como demonstra o exemplo acima (supondo que todas são células de avaliações conhecidas), é possível visualizar que a matriz original (representada por "X-Original") possui valores 1, 2, 3 em seus itens $A1$, $A2$, $A3$. E 4, 5, 6 em $B1$, $B2$, $B3$, todos respectivamente.

Também observamos que a matriz estimada ("X-Estimada") tenha os valores de 7, 8, 9 em $A1$, $A2$, $A3$, e 10, 11, 12 em $B1$, $B2$, $B3$, todos respectivamente. Então para encontrar o "*li* " de cada um desses dois usuários seria:

$$((A1 - A1)^2 + (A2 - A2)^2 + (A3 - A3)^2)/3$$
$$= ((7 - 1)^2 + (8 - 2)^2 + (9 - 3)^2)/3$$

&

$$((B1 - B1)^2 + (B2 - B2)^2 + (B3 - B3)^2)/3$$
$$= ((10 - 4)^2 + (11 - 5)^2 + (12 - 6)^2)/3$$

A divisão é representada pela quantidade de células comparadas, se tivéssemos $A4$, $A5$, por exemplo, seria divido por 4 ou 5 respectivamente.

# APÊNDICE B

# INSERÇÕES NO CÓDIGO DE VIANNA

## CLASSE RPOLCONTROLLER

Classe: com.srh.api.controller.RpolController

Algoritmo B.1: Código da classe RpolController.

```
1  package com.srh.api.controller;
2
3  import com.srh.api.dto.resource.ProjectDto;
4  import com.srh.api.hypermedia.ProjectModelAssembler;
5  import com.srh.api.model.Project;
6  import com.srh.api.service.ProjectService;
7  import com.srh.api.service.RpolService;
8  import org.springframework.beans.factory.annotation.Autowired;
9  import org.springframework.data.web.PagedResourcesAssembler;
10 import org.springframework.web.bind.annotation.*;
11
12 import java.util.ArrayList;
13
14 @RestController
15 @RequestMapping("/rpols")
16 public class RpolController {
17     @Autowired
18     private RpolService rpolService;
19
20     @Autowired
21     private ProjectService projectService;
22
23     @GetMapping("/{ProjectId}/{AlgorithmId}")
24     public Double findARpol(@PathVariable Integer ProjectId,
          @PathVariable Integer AlgorithmId) {
```

```
25          Project project = projectService.find(ProjectId);
26          if (project.getId() == ProjectId) {
27              Double rpol = rpolService.getRpol(ProjectId,
                    AlgorithmId);
28              return rpol;
29          }
30          return 0.0;
31      }
32
33  }
```

Listing B.1 - Código da classe RpolController

## CLASSE RPOLSERVICE

Classe: com.srh.api.service.RpolService

Algoritmo B.2: Código da classe RpolService.

```
1  package com.srh.api.service;
2
3  import com.srh.api.model.*;
4  import com.srh.api.repository.ItemRatingRepository;
5  import com.srh.api.repository.ItemRepository;
6  import com.srh.api.repository.ProjectRepository;
7  import com.srh.api.repository.RecommendationRepository;
8
9  import java.util.ArrayList;
10
11 import org.springframework.beans.factory.annotation.Autowired;
12 import org.springframework.stereotype.Service;
13
14 @Service
15 public class RpolService {
16
```

```
17      @Autowired
18      private ProjectRepository projectRepository;
19
20      @Autowired
21      private RecommendationRepository recommendationRepository;
22
23      @Autowired
24      private ItemRatingRepository itemRatingRepository;
25
26      @Autowired
27      private ItemRepository itemRepository;
28
29      public Double getRpol(Integer ProjectId, Integer AlgorithmId) {
30
31          ArrayList<Double> rpol = new ArrayList<>();
32          double auxRpol = 0;
33
34          ArrayList<Double> varItem = new ArrayList<>();
35          double auxVarItem = 0;
36
37          double totalVar = 0;
38          int qtdAvaliacao = 0;
39          double totalItem = 0;
40          double mediaItem = 0;
41
42          ArrayList<Double> listaAlgoritmo1 = new ArrayList<>();
43          ArrayList<Double> listaAlgoritmo2 = new ArrayList<>();
44          boolean a1 = false;
45          boolean a2 = false;
46          boolean a4 = false;
47
48          ArrayList<Double> sublista = null;
49
50          Iterable<ItemRating> lista1 =
51                  itemRatingRepository.findAll();
```

```
52          Iterable<Recommendation> lista2 =
53                  recommendationRepository.findAll();
54          Iterable<Item> itens = itemRepository.findAll();
55
56          if (AlgorithmId == 4) {
57              for (Item i : itens) {
58                  sublista = new ArrayList();
59                  for (ItemRating ir : lista1) {
60                      if (ir.getId().getItem().getId() == i.getId())
                            {
61                          sublista.add(ir.getScore());
62                      }
63                  }
64                  for (Recommendation r : lista2) {
65                      if (r.getAlgorithm().getId() == 1 &&
66                              r.getItem().getId() == i.getId()) {
67                          listaAlgoritmo1.add(r.getWeight());
68                          a1 = true;
69                      }
70                      if (r.getAlgorithm().getId() == 2 &&
71                              r.getItem().getId() == i.getId()) {
72                          listaAlgoritmo2.add(r.getWeight());
73                          a2 = true;
74                      }
75                      if (a1 == true && a2 == true) {
76                          a4 = true;
77                      }
78                  }
79                  double menor = 0;
80                  for (int x = 0; x < listaAlgoritmo1.size(); x++) {
81                      if (a4 == true) {
82                          if (listaAlgoritmo1.get(x).doubleValue() <
83                                  listaAlgoritmo2.get(x).doubleValue
                                        ()) {
84                              menor = listaAlgoritmo1.get(x);
```

```
85                              sublista.add(menor);
86                          } else {
87                              menor = listaAlgoritmo2.get(x);
88                              sublista.add(menor);
89                          }
90                      }
91                  }
92                  if (sublista.size() != i.getProject().getEvaluators
                        ().size()) {
93                      sublista.clear();
94                      listaAlgoritmo1.clear();
95                      listaAlgoritmo2.clear();
96                      a4 = false;
97                  }
98                  for (Double d : sublista) {
99                      totalItem = totalItem + d;
100                     qtdAvaliacao++;
101                 }
102                 mediaItem = totalItem / qtdAvaliacao;
103
104                 for (Double sub : sublista) {
105                     auxVarItem = auxVarItem + (Math.pow(sub -
                            mediaItem, 2));
106                 }
107                 auxVarItem = auxVarItem / qtdAvaliacao;
108
109                 varItem.add(auxVarItem);
110
111                 totalVar = totalVar + auxVarItem;
112
113                 auxRpol = totalVar / varItem.size();
114
115                 totalItem = 0;
116                 qtdAvaliacao = 0;
117                 mediaItem = 0;
```

```
118                 auxVarItem = 0;
119                 listaAlgoritmo1.clear();
120                 listaAlgoritmo2.clear();
121                 a1 = false;
122                 a2 = false;
123                 a4 = false;
124             }
125 
126         } else {
127             for (Item i : itens) {
128 
129                 sublista = new ArrayList();
130 
131                 for (ItemRating ir : lista1) {
132                     if (ir.getId().getItem().getId() == i.getId())
                        {
133                         sublista.add(ir.getScore());
134                     }
135                 }
136                 for (Recommendation r : lista2) {
137                     if (r.getItem().getId() == i.getId() &&
138                             r.getAlgorithm().getId() == AlgorithmId
                                ) {
139                         sublista.add(r.getWeight());
140                     }
141                 }
142 
143                 if (sublista.size() != i.getProject().getEvaluators
                    ().size()) {
144                     sublista.clear();
145                 }
146 
147                 for (Double d : sublista) {
148                     totalItem = totalItem + d;
149                     qtdAvaliacao++;
```

```
150                 }
151
152                 mediaItem = totalItem / qtdAvaliacao;
153
154                 for (Double sub : sublista) {
155                     auxVarItem = auxVarItem + (Math.pow(sub -
                           mediaItem, 2));
156                 }
157                 auxVarItem = auxVarItem / qtdAvaliacao;
158
159                 varItem.add(auxVarItem);
160
161                 totalVar = totalVar + auxVarItem;
162
163                 auxRpol = totalVar / varItem.size();
164
165                 totalItem = 0;
166                 qtdAvaliacao = 0;
167                 mediaItem = 0;
168                 auxVarItem = 0;
169             }
170         }
171         rpol.add(auxRpol);
172         sublista.clear();
173         listaAlgoritmo1.clear();
174         listaAlgoritmo2.clear();
175         a1 = false;
176         a2 = false;
177         a4 = false;
178         varItem.clear();
179         totalItem = 0;
180         totalVar = 0;
181         qtdAvaliacao = 0;
182         mediaItem = 0;
183         auxVarItem = 0;
```

```
184
185             return auxRpol;
186         }
187 }
```

Listing B.2 - Código da classe RpolService

## CLASSE RINDVCONTROLLER

Classe: com.srh.api.controller.RindvController

Algoritmo B.3: Código da classe RindvController.

```
1  package com.srh.api.controller;
2
3  import com.srh.api.model.Project;
4  import com.srh.api.service.ProjectService;
5  import com.srh.api.service.RindvService;
6  import org.springframework.beans.factory.annotation.Autowired;
7  import org.springframework.web.bind.annotation.GetMapping;
8  import org.springframework.web.bind.annotation.PathVariable;
9  import org.springframework.web.bind.annotation.RequestMapping;
10 import org.springframework.web.bind.annotation.RestController;
11
12 import java.util.ArrayList;
13
14 @RestController
15 @RequestMapping("/rindvs")
16 public class RindvController {
17
18     @Autowired
19     private RindvService rindvService;
20
21     @Autowired
22     private ProjectService projectService;
```

```
23
24      @GetMapping("/{ProjectId}/{AlgorithmId}")
25      public Double findARindv(@PathVariable Integer ProjectId,
            @PathVariable Integer AlgorithmId) {
26          Project project = projectService.find(ProjectId);
27          if (project.getId() == ProjectId) {
28              Double rindv = rindvService.getRindv(ProjectId,
                    AlgorithmId);
29              return rindv;
30          }
31          return 0.0;
32      }
33
34  }
```

Listing B.3 – Código da classe RindvController

## CLASSE RINDVSERVICE

Classe: com.srh.api.service.RindvService

Algoritmo B.4: Código da classe RindvService.

```
1  package com.srh.api.service;
2
3  import com.srh.api.model.*;
4  import com.srh.api.repository.ProjectRepository;
5  import com.srh.api.repository.RecommendationRatingRepository;
6  import com.srh.api.repository.RecommendationRepository;
7  import org.springframework.beans.factory.annotation.Autowired;
8  import org.springframework.stereotype.Service;
9
10 import java.util.ArrayList;
11
12
```

```
13 @Service
14 public class RindvService {
15
16     @Autowired
17     private ProjectRepository projectRepository;
18
19     @Autowired
20     private RecommendationRepository recommendationRepository;
21
22     @Autowired
23     private RecommendationRatingRepository
           recommendationRatingRepository;
24
25     public Double getRindv(Integer ProjectId, Integer AlgorithmId)
           {
26
27         ArrayList<Double> totaisMatrizes = new ArrayList<>();
28         Double mediaLi;
29         ArrayList<Double> liUser = new ArrayList<>();
30         double auxli = 0;
31         ArrayList<Double> rindv = new ArrayList<>();
32         double auxRindv = 0;
33
34         double totalItem = 0;
35         double auxMediaLi = 0;
36         int xComparacao = 0;
37         int qtdScores = 0;
38
39         Iterable<Recommendation> lista1 = recommendationRepository.
               findAll();
40         Iterable<RecommendationRating> lista2 =
               recommendationRatingRepository.findAll();
41
42         ArrayList<Double> listaAlgoritmo4 = new ArrayList<>();
43         for (Recommendation r : lista1) {
```

```
44              for (Recommendation r2 : lista1) {
45                  if (r.getAlgorithm().getId() == 1 &&
46                          r2.getAlgorithm().getId() == 2 &&
47                          r.getEvaluator().getId() == r2.getEvaluator
                            ().getId() &&
48                          r.getItem().getId() == r2.getItem().getId()
                            ) {
49                      if (r.getWeight().doubleValue() < r2.getWeight
                        ().doubleValue()) {
50                          listaAlgoritmo4.add(r.getWeight());
51                      } else {
52                          listaAlgoritmo4.add(r2.getWeight());
53                      }
54                  }
55              }
56          }
57
58          if (AlgorithmId == 4) {
59              for (int x = 0; x < listaAlgoritmo4.size(); x++) {
60                  for (RecommendationRating irr : lista2) {
61                      if (irr.getId() == x + 1 &&
62                              xComparacao < 2 &&
63                              qtdScores < listaAlgoritmo4.size()) {
64                          totalItem = totalItem + (listaAlgoritmo4.
                            get(x) + irr.getScore());
65                          auxli = auxli + Math.pow(listaAlgoritmo4.
                            get(x) - irr.getScore(), 2);
66                          xComparacao++;
67                          qtdScores++;
68                      }
69                  }
70                  if (xComparacao >= 2) {
71                      auxli = auxli / xComparacao;
72                      liUser.add(auxli);
73                      auxli = 0;
```

```
74                      totaisMatrizes.add(totalItem);
75                      totalItem = 0;
76                      xComparacao = 0;
77                  }
78              }
79          } else {
80              for (Recommendation r : lista1) {
81                  for (RecommendationRating irr : lista2) {
82                      if (r.getEvaluator().getId() == irr.
                            getEvaluator().getId() &&
83                              irr.getRecommendation().getId() == r.
                                  getId() &&
84                              r.getAlgorithm().getId() == AlgorithmId
                                  ) {
85                          totalItem = totalItem + (r.getWeight() +
                                irr.getScore());
86                          auxli = auxli + Math.pow(r.getWeight() -
                                irr.getScore(), 2);
87                          xComparacao++;
88                      }
89                  }
90                  if (xComparacao >= 2) {
91                      auxli = auxli / xComparacao;
92                      liUser.add(auxli);
93                      auxli = 0;
94                      totaisMatrizes.add(totalItem);
95                      totalItem = 0;
96                      xComparacao = 0;
97                  }
98              }
99          }
100
101         for (int i = 0; i < liUser.size(); i++) {
102             auxMediaLi = auxMediaLi + liUser.get(i);
103         }
```

```
104         mediaLi = auxMediaLi / liUser.size();
105
106         for (Double lis : liUser) {
107             auxRindv = auxRindv + (Math.pow(lis - mediaLi, 2));
108         }
109         auxRindv = auxRindv / liUser.size();
110
111         rindv.add(auxRindv);
112
113         mediaLi = 0.0;
114         totalItem = 0.0;
115         auxMediaLi = 0.0;
116         liUser.clear();
117         auxli = 0;
118         qtdScores = 0;
119
120         return auxRindv;
121     }
122 }
```

Listing B.4 - Código da classe RindvService

**CLASSE RGRPCONTROLLER Classe:**

com.srh.api.controller.RgrpController

Algoritmo B.5: Código da classe RgrpController.

```
1 package com.srh.api.controller;
2
3 import com.srh.api.model.Project;
4 import com.srh.api.service.ProjectService;
5 import com.srh.api.service.RgrpService;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.web.bind.annotation.GetMapping;
```

```
8  import org.springframework.web.bind.annotation.PathVariable;
9  import org.springframework.web.bind.annotation.RequestMapping;
10 import org.springframework.web.bind.annotation.RestController;
11
12 import java.util.ArrayList;
13
14 @RestController
15 @RequestMapping("/rgrps")
16 public class RgrpController {
17
18     @Autowired
19     private RgrpService rgrpService;
20
21     @Autowired
22     private ProjectService projectService;
23
24     @GetMapping("/{ProjectId}/{AlgorithmId}")
25     public Double findARindv(@PathVariable Integer ProjectId,
           @PathVariable Integer AlgorithmId) {
26         Project project = projectService.find(ProjectId);
27         if (project.getId() == ProjectId) {
28             Double rgrp = rgrpService.getRgrp(ProjectId,
                   AlgorithmId);
29             return rgrp;
30         }
31         return 0.0;
32     }
33
34 }
```

Listing B.5 - Código da classe RgrpController

## CLASSE RGRPSERVICE

Classe: com.srh.api.service.RgrpService

Algoritmo B.6: Código da classe RgrpService.

```
1  package com.srh.api.service;
2
3
4  import com.srh.api.model.Recommendation;
5  import com.srh.api.model.RecommendationRating;
6  import com.srh.api.repository.ProjectRepository;
7  import com.srh.api.repository.RecommendationRatingRepository;
8  import com.srh.api.repository.RecommendationRepository;
9  import org.springframework.beans.factory.annotation.Autowired;
10 import org.springframework.stereotype.Service;
11
12 import java.util.ArrayList;
13
14
15 @Service
16 public class RgrpService {
17
18     @Autowired
19     private ProjectRepository projectRepository;
20
21     @Autowired
22     private RecommendationRepository recommendationRepository;
23
24     @Autowired
25     private RecommendationRatingRepository
          recommendationRatingRepository;
26
27     public Double getRgrp(Integer ProjectId, Integer AlgorithmId) {
28
29         ArrayList<Double> grupo1;
30         ArrayList<Double> grupo2;
31         grupo1 = new ArrayList<>();
32         grupo2 = new ArrayList<>();
```

```
33
34          ArrayList<Double> liUser = new ArrayList<>();
35          double auxli = 0;
36          ArrayList<Double> lIUser = new ArrayList<>();
37          double auxLI = 0;
38          ArrayList<Double> rgrp = new ArrayList<>();
39          double auxRgrp = 0;
40
41          double mediaLI = 0;
42          double totalItem = 0;
43          int xComparacao = 0;
44          int grp = 0;
45          int qtdScores = 0;
46
47          Iterable<Recommendation> lista1 =
48                  recommendationRepository.findAll();
49          Iterable<RecommendationRating> lista2 =
50                  recommendationRatingRepository.findAll();
51
52          ArrayList<Double> listaAlgoritmo4 = new ArrayList<>();
53          for (Recommendation r : lista1) {
54              for (Recommendation r2 : lista1) {
55                  if (r.getAlgorithm().getId() == 1 &&
56                          r2.getAlgorithm().getId() == 2 &&
57                          r.getEvaluator().getId() == r2.getEvaluator
                            ().getId() &&
58                          r.getItem().getId() == r2.getItem().getId()
                            ) {
59                      if (r.getWeight().doubleValue()
60                              < r2.getWeight().doubleValue()) {
61                          listaAlgoritmo4.add(r.getWeight());
62                      } else {
63                          listaAlgoritmo4.add(r2.getWeight());
64                      }
65                  }
```

```
66              }
67          }
68
69          if (AlgorithmId == 4) {
70              for (int x = 0; x < listaAlgoritmo4.size(); x++) {
71                  for (RecommendationRating irr : lista2) {
72                      if (irr.getId() == x + 1 &&
73                              xComparacao < 2 &&
74                              qtdScores < listaAlgoritmo4.size()) {
75                          totalItem = totalItem + (listaAlgoritmo4.
                             get(x)
76                                  + irr.getScore());
77                          auxli = auxli + Math.pow(listaAlgoritmo4.
                             get(x)
78                                  - irr.getScore(), 2);
79                          xComparacao++;
80                          qtdScores++;
81                      }
82                  }
83                  if (xComparacao >= 2) {
84                      auxli = auxli / xComparacao;
85                      liUser.add(auxli);
86                      auxli = 0;
87                      totalItem = 0;
88                      xComparacao = 0;
89                      qtdScores = 0;
90                  }
91              }
92
93              for (int x = 0; x < liUser.size(); x++) {
94                  if (x % 2 == 1) {
95                      grupo1.add(liUser.get(x));
96                  } else {
97                      grupo2.add(liUser.get(x));
98                  }
```

```
99              }
100         } else {
101             for (Recommendation r : lista1) {
102                 for (RecommendationRating irr : lista2) {
103                     if (r.getEvaluator().getId() == irr.
                            getEvaluator().getId() &&
104                             irr.getRecommendation().getId() == r.
                                    getId() &&
105                             r.getAlgorithm().getId() == AlgorithmId
                                    ) {
106                         totalItem = totalItem + (r.getWeight() +
                                irr.getScore());
107                         auxli = auxli + Math.pow(r.getWeight() -
                                irr.getScore(), 2);
108                         xComparacao++;
109                     }
110                 }
111                 if (xComparacao >= 2) {
112                     auxli = auxli / xComparacao;
113                     grp = r.getEvaluator().getId();
114                     if (grp % 2 == 0) {
115                         grupo2.add(auxli);
116                         liUser.add(auxli);
117                         auxli = 0;
118                         totalItem = 0;
119                         xComparacao = 0;
120                         grp = 0;
121                     }
122                     if (grp % 2 == 1) {
123                         grupo1.add(auxli);
124                         liUser.add(auxli);
125                         auxli = 0;
126                         totalItem = 0;
127                         xComparacao = 0;
128                         grp = 0;
```

```
129                         }
130                     }
131                 }
132             }
133
134             for (int i = 0; i < grupo2.size(); i++) {
135                 auxLI = auxLI + grupo2.get(i).doubleValue();
136             }
137             auxLI = auxLI / grupo2.size();
138             lIUser.add(auxLI);
139
140             auxLI = 0;
141             for (int i = 0; i < grupo1.size(); i++) {
142                 auxLI = auxLI + grupo1.get(i).doubleValue();
143             }
144             auxLI = auxLI / grupo1.size();
145             lIUser.add(auxLI);
146
147             auxLI = 0;
148             for (int i = 0; i < lIUser.size(); i++) {
149                 auxLI = auxLI + lIUser.get(i);
150             }
151             mediaLI = auxLI / lIUser.size();
152
153             for (Double lIs : lIUser) {
154                 auxRgrp = auxRgrp + (Math.pow(lIs - mediaLI, 2));
155             }
156             auxRgrp = auxRgrp / lIUser.size();
157
158             rgrp.add(auxRgrp);
159
160             liUser.clear();
161             auxLI = 0.0;
162             mediaLI = 0.0;
163             lIUser.clear();
```

```
164             grupo2.clear();
165             grupo1.clear();
166             qtdScores = 0;
167 
168             return auxRgrp;
169         }
170 }
```

Listing B.6 - Código da classe RgrpService

# ANEXO A – ANEXOS DOS EXEMPLOS

Para visualização da planilha dos exemplos citados na metodologia:

- Planilha do exemplo - Metodologia

GitHub dos códigos fonte aplicados no exemplo:

- https://github.com/marciocgl/Rpol
- https://github.com/marciocgl/Rindv
- https://github.com/marciocgl/Rgrp

# ANEXO B

## RESULTADOS DETALHADOS DAS MEDIDAS

Para visualização dos resultados de forma mais detalhada sobre as avaliações do caso de uso, e das recomendações dos itens para os avaliadores:

- Anexo dos resultados


ISBN 978-655376311-1
9 786553 763111