MEC-SETEC
INSTITUTO FEDERAL MINAS GERAIS - *Campus* Formiga
Curso de Ciência da Computação

FILTRAGEM HÍBRIDA PARA SISTEMA DE RECOMENDAÇÃO DE LIVROS UTILIZANDO REDES NEURAIS

BRUNA CRISTINA MENDES

Orientador: Everthon Valadão

Formiga - MG

2023

BRUNA CRISTINA MENDES

FILTRAGEM HÍBRIDA PARA SISTEMA DE RECOMENDAÇÃO DE LIVROS UTILIZANDO REDES NEURAIS

Monografia do trabalho de conclusão de curso apresentado ao Instituto Federal Minas Gerais - Campus Formiga, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Everthon Valadão

Formiga - MG

2023

Mendes, Bruna Cristina
M538f Filtragem híbrida para sistema de recomendação de livros utilizando redes neurais / Bruna Cristina Mendes – Formiga : IFMG, 2023.
64p. : il. color.

Orientador: Prof. Me. Everthon Valadão dos Santos
Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) Instituto Federal de Educação,Ciência e Tecnologia de Minas Gerais – *Campus* Formiga.

1. Sistema de recomendação. 2. Redes neurais profundas (DNN). 3. Filtragem híbrida. 4. BookCrossing. I. Santos, Everthon Valadão dos. II. Título.

CDD 004

Ficha catalográfica elaborada pela Bibliotecária Lívia Renata Santos- CRB/6-2561

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS
Campus Formiga
Diretoria de Ensino
Docência Área Acadêmica de Computação
Rua São Luiz Gonzaga, s/n - Bairro São Luiz - CEP 35570-000 - Formiga - MG
- www.ifmg.edu.br

BRUNA CRISTINA MENDES

Filtragem híbrida para sistema de recomendação de livros utilizando redes neurais

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Minas Gerais - Campus Formiga, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

APROVADO em: 19 de dezembro de 2023.

BANCA EXAMINADORA

Prof.º Me. Everthon Valadão (orientador, IFMG)

Prof.º Dr. Alexandre Pimenta (IFMG)

Prof.º Dr. Bruno Ferreira (IFMG)

Documento assinado eletronicamente por **Everthon Valadao dos Santos**, **Professor**, em 19/12/2023, às 14:48, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Bruno Ferreira**, **Professor**, em 19/12/2023, às 14:48, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Alexandre Pimenta**, **Professor**, em 19/12/2023, às 14:50, conforme Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade do documento pode ser conferida no site https://sei.ifmg.edu.br/consultadocs informando o código verificador **1776475** e o código CRC **435BCE73**.

Agradecimentos

Agradeço primeiramente à minha família, em especial aos meus pais, Eduardo e Danielli, pela força e incentivo ao longo desses anos, eu não teria conseguido concluir o curso sem o apoio incondicional deles.

Agradeço ao meu orientador Everthon Valadão pelo auxílio, paciência e orientação nesta pesquisa. Foi um processo longo e cheio de obstáculos, e eu fico muito grata que ele tenha me ajudado até o final. Também a todos os amigos que fiz no curso de graduação e que estiveram ao meu lado para enfrentar tantos desafios, e que também estarão no futuro.

Por fim, agradeço ao Instituto Federal de Minas Gerais Campus Formiga, ao corpo docente e a todos os meus colegas de curso que me acompanharam nos cinco anos que estivemos juntos.

You have to pretend you get an endgame.
You have to carry on like you will; otherwise,
you can't carry on at all.
(Rainbow Rowell, Carry on)

Resumo

Esta monografia aborda o desenvolvimento de um sistema de recomendação híbrido para livros, combinando abordagens colaborativas e baseadas em conteúdo, impulsionado por Redes Neurais. A crescente complexidade e diversidade dos dados tornam essencial a adoção de modelos híbridos, e este trabalho justifica-se pela necessidade de oferecer aos leitores sugestões mais relevantes e personalizadas, superando as limitações dos modelos tradicionais. O objetivo do trabalho é estudar, implementar e avaliar um sistema de recomendação híbrido, utilizando uma *Deep Neural Network* (DNN), que não apenas sugira obras com base em preferências passadas, mas também leve em conta características literárias específicas, proporcionando recomendações mais contextuais. A validação foi conduzida por meio de experimentos utilizando os conjuntos de dados do BookCrossing e da Amazon, e a avaliação é feita comparando a precisão das recomendações com o modelo de filtragem colaborativa e identificando melhorias na personalização do modelo híbrido proposto. As principais contribuições deste trabalho incluem a implementação de um sistema híbrido eficaz, que possibilita a avaliação comparativa com modelos tradicionais e análise das melhorias alcançadas.

Palavras-chave: Sistema de Recomendação, Redes Neurais Profundas (DNN), Filtragem Híbrida, BookCrossing.

Abstract

This thesis addresses the development of a hybrid recommendation system for books, combining collaborative and content-based approaches, powered by Neural Networks. The increasing complexity and diversity of data make adopting hybrid models essential, and this work is justified by the need to provide readers with more relevant and personalized suggestions, overcoming the limitations of traditional models. The study aims to explore, implement, and evaluate a hybrid recommendation system using a *Deep Neural Network* (DNN), which not only suggests works based on past preferences but also takes into account specific literary characteristics, providing more contextual recommendations. Validation was conducted through experiments using the BookCrossing and Amazon datasets, with the evaluation comparing the accuracy of recommendations with the collaborative model and identifying improvements in the personalization of the proposed hybrid model. The main contributions of this work include implementing an effective hybrid system, enabling comparative evaluation with traditional models, and analysis of the improvements achieved.

Keywords: Recommender System, *Deep Neural Network* (DNN), Hybrid Filtering, BookCrossing.

Lista de ilustrações

Lista de tabelas

Lista de quadros

Lista de abreviaturas e siglas

API	Interface de Programação de Aplicações
CNN	*Convolutional Neural Network*
DNN	*Deep Neural Network*
FC	Filtragem Colaborativa
FBC	Filtragem Baseada em Conteúdo
kNN	*k-Nearest Neighbours*
LDA	*Linear Discriminant Analysis*
MF	*Matrix Factorization*
MAE	Erro Absoluto Médio
MSE	Erro Quadrático Médio
OM	*Opinion Mining*
PCA	*Principal Component Analysis*
ReLU	*rectified-linear-units*
RMSE	Raiz do Erro Quadrático Médio
RNA	Rede Neural Artificial
RNN	*Recurrent Neural Network*
SPM	*Sequential Pattern Mining*
SR	Sistemas de Recomendação
SVD	*Singular Value Decomposition*
SGD	Gradiente Descendente Estocástico

Sumário

1 INTRODUÇÃO

Sistemas de Recomendação (SR) são largamente utilizados para sugerir filmes, músicas, vídeos, produtos online, e diversos outros itens. Eles aceleram o processo de busca do usuário ao encontrarem entidades relevantes e relacionados a preferências identificadas a partir de algum comportamento apresentado, e frequentemente usam essa informação para maximizar vendas ou gerar mais interesse nos conteúdos oferecidos (PU; CHEN; HU, 2012). Esses sistemas possuem o papel de filtrar as informações mais importantes do usuário a partir de uma abundância de dados disponíveis na internet e fornecer sugestões de itens personalizados, como os livros recomendados pela Amazon[1], os filmes e séries recomendados pela NETFLIX[2] e as músicas pelo Spotify[3].

Os SR estão preocupados em coletar dados, processá-los e extrair o máximo de informações usando métodos estatísticos e analíticos. Eles surgiram na década de 1990 e abordagens bidimensionais (usuários e itens) eram predominantemente usados para prever interesses de usuários (ADOMAVICIUS; TUZHILIN, 2001).

Um recomendador de livros está inserido no mesmo contexto de um recomendador de produtos em websites de *e-commerce*. Consumidores frequentemente se encontram diante de uma gama de produtos e informações das quais podem escolher, e muitos websites incorporam sistemas de recomendação que oferecem ao usuário uma lista de itens ou páginas que possuem maior chance de serem interessantes.

Muitos sistemas de recomendação de livros já foram implementados, dentre os quais: Linden, Smith e York (2003) e Xin et al. (2013). Entretanto, estes possuem capacidades limitadas de recomendar livros que conseguem atender as escolhas e necessidades do usuário. A maioria deles apenas considera a opinião de outros usuários sobre livros semelhantes, ou apenas processam as descrições

[1] https://www.amazon.com.br/. Acesso em: 20 dez. 2023.
[2] https://www.netflix.com/. Acesso em: 20 dez. 2023.
[3] https://www.spotify.com/. Acesso em: 20 dez. 2023.

e características dos livros, com base no conteúdo. Um exemplo de sistema de recomendação de livros é o mecanismo do site da Amazon, que consegue filtrar milhões de itens disponíveis baseado nas preferências ou histórico de navegação dos usuários.

Este trabalho propõe a criação de um modelo para um sistema de recomendação direcionado a leitores que procuram comprar livros pela internet. O objetivo da pesquisa é evidenciar as principais abordagens atualmente disponíveis, como filtragem por conteúdo e filtragem colaborativa, com técnicas de mineração de dados e aprendizado de máquina, para então aplicar a melhor solução a fim produzir recomendações de livros mais eficientes e relevantes. Para tal objetivo, será desenvolvida uma abordagem híbrida utilizando Rede Neural Artificial (RNA), que irá combinar informações de usuários e informações categóricas de livros a fim de solucionar problemas comuns encontrados em métodos tradicionais.

1.1 Justificativa

O desenvolvimento de sistemas de recomendação de livros é um esforço multidisciplinar que envolve conhecimento de vários campos, como inteligência artificial, mineração de dados, estatística, sistemas de apoio à decisão, marketing e comportamento de consumidores.

Nos anos recentes, houve um aumento exponencial no volume de informações digitais disponíveis, fontes eletrônicas e serviços online. A sobrecarga de informações criou o problema de como filtrar e entregar uma informação relevante para o usuário.

A motivação para a realização deste projeto partiu do interesse de encontrar métodos diferentes para resolver as lacunas dos sistemas de recomendação de livros existentes no mercado. Dentre os mais conhecidos, sabe-se que a Amazon recomenda livros que foram comprados e avaliados por clientes que apresentaram preferências similares ao usuário para quem a recomendação é feita. Dessa forma, esse método pode ser usado sem informações efetivas sobre o livro em si, desde que as avaliações *(feedback)* dos usuários estejam disponíveis (KUROIWA; BHALLA, 2007). Entretanto, essa abordagem pode não ser a mais eficaz para recomendar livros impopulares que leitores não compram ou avaliam.

Muitos trabalhos têm sido feito para a implementação desses sistemas, visto que a quantidade de setores que apresentam problemas que podem ser resolvidos por esses algoritmos. Ademais, existe uma quantidade infinita de possibilidades nesse campo de pesquisa, que podem ser aplicadas tanto academicamente quanto comercialmente.

1.2 Objetivos

1.2.1 Objetivo Geral

Criar um sistema de recomendação personalizado para livros no contexto de sites de vendas online, utilizando uma Rede Neural Artificial (RNA), mais especificamente uma DNN, em uma abordagem híbrida para obter um bom desempenho preditivo de livros mais relevantes para um usuário.

1.2.2 Objetivos Específicos

1. Estudar os principais métodos utilizados em Sistemas de Recomendação e determinar a melhor abordagem híbrida, com foco especial na incorporação de informações categóricas de livros no método escolhido.

2. Identificar e obter as principais bases de dados para treinamento e testes, considerando a presença de informações categóricas e a necessidade de incorporá-las na solução proposta.

3. Pré-processar dados e selecionar as informações que serão usadas no modelo segundo a abordagem de filtragem híbrida escolhida, que envolve a combinação de dois ou mais métodos de filtragem;

4. Modelar e treinar uma *Deep Neural Network* (DNN) para prever as avaliações dos usuários e produzir recomendações, dando destaque à utilização eficaz das informações categóricas de livros durante o treinamento.

5. Quantificar a eficácia do modelo implementado usando métricas de avaliação que busquem a minimização do erro das previsões calculadas.

1.3 Estrutura do trabalho

Este trabalho é composto por seis capítulos, sendo este primeiro onde foi feita a contextualização da pesquisa. No Capítulo 2, são apresentados os conceitos teóricos necessários para o entendimento do trabalho. O Capítulo 3 apresenta todas as tecnologias necessárias para a implementação da solução proposta, como linguagens de programação, bibliotecas e conjuntos de dados. No Capítulo 4 são descritas as etapas realizadas no desenvolvimento do projeto e como foi conduzida a realização do experimento. No Capítulo 5, discute-se os resultados dos testes feitos, de forma que são analisados e comentados. Por fim, no Capítulo 6 são apresentadas as considerações finais e propostas para possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo traz o levantamento bibliográfico e os principais conceitos necessários para a compreensão e desenvolvimento deste trabalho de conclusão de curso, tais como a teoria por trás dos sistemas de recomendação e o atual estado da arte, as métricas de avaliação, e redes neurais. Então, ao final do capítulo, serão apresentados alguns trabalhos relacionados.

2.1 Sistemas de Recomendação

Com o aumento da quantidade de informações disponibilizadas com o auxílio da Internet, redes sociais e aplicativos móveis, a dificuldade para usuários encontrarem conteúdos relevantes aumentam proporcionalmente. Desta forma, surge a necessidade de identificar conteúdos que interessam ao usuário em meio a uma grande variedade de tópicos e itens.

Pode-se dizer que Sistemas de Recomendação (SR) é uma técnica de filtragem de informação personalizada, usada para predizer quais ou quantos itens um usuário pode se interessar. Ela explora a relação entre usuários e itens, a relação de usuários entre si ou itens entre si, e o objetivo principal é fornecer aos usuários as melhores recomendações. Se o item oferecido ao usuário tiver alta probabilidade de ser utilizado, então a sugestão é considerada relevante (NOGUEIRA et al., 2015).

Os SR são normalmente classificados conforme os diferentes tipos de informação disponibilizados pela base de dados. Dentre os mais referenciados na literatura Burke (2002), Aggarwal et al. (2016) e Adomavicius e Tuzhilin (2005), estão:

(i) **Filtragem Colaborativa (FC)**, que faz uso da similaridade de interesses entre usuários, calculada a partir da semelhança do histórico de interações com itens - como avaliações (notas), histórico de compras, etc - entre o usuário ativo e outras pessoas com preferências relacionadas. Essa é a razão pela qual os autores de Schafer, Konstan e Riedl (2001) refere-se a filtragem colaborativa como "correlação

de pessoa para pessoa";

(ii) **Filtragem Baseada em Conteúdo (FBC)**, que usa informações dos atributos dos itens para produzir recomendações, tais como: palavras-chave, categorias, descrição textual, entre outros. As recomendações são feitas a partir do perfil do usuário, ou seja, as preferências do próprio usuário registradas no passado;

(iii) **Filtragem Híbrida**, que procura combinar dois ou mais tipos de filtragem para aumentar a eficácia ao superar as limitações dos métodos individuais.

2.1.1 Filtragem Colaborativa

De acordo com Sohail, Anwar e Siddiqui (2019), a Filtragem Colaborativa (FC) é o tipo mais utilizado para implementar sistemas de recomendação desde o aparecimento do primeiro sistema na metade da década de 1990. O método tradicional propõe analisar o histórico comportamental de um usuário alvo (*i.e.* cliente para quem a recomendação é feita), encontrar outros usuários com opiniões similares, e usar essas opiniões para gerar uma lista de recomendação.

Segundo Aggarwal et al. (2016), existem dois tipos de abordagens comumente utilizadas em Filtragem Colaborativa: baseado em vizinhança e em modelo.

2.1.1.1 Baseado em vizinhança

Também conhecido como Filtragem Colaborativa Baseada em Memória, esses algoritmos partem da premissa que usuários similares possuem padrões de comportamento similares, e por isso, itens similares recebem notas parecidas.

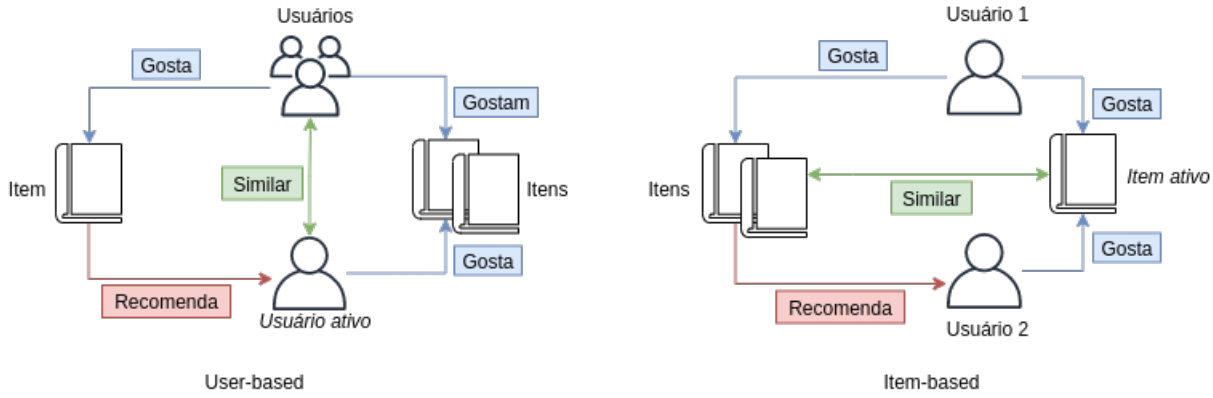
Nessa abordagem, os dados são armazenados em memória e são calculadas medidas de similaridade entre itens e/ou entre usuários, a cada vez que é solicitada alguma recomendação (NOGUEIRA et al., 2015).

Existem dois tipos primários: podem ser **baseados em usuários** (*user-based*) ou **baseados em itens** (*item-based*). No primeiro caso, a abordagem *user-based* (WANG; VRIES; REINDERS, 2006) identifica uma vizinhança de usuários que são semelhantes ao *usuário ativo* (i.e., ou avaliam itens diferentes de forma similar, ou tendem a comprar conjuntos de itens semelhantes). Este conjunto de

vizinhos é baseado na semelhança das preferências observadas entre esses usuários e o usuário ativo. Em seguida, esses sistemas usam algoritmos que combinam essas preferências a fim de produzir uma lista de recomendações para o usuário.

A abordagem *item-based* (LINDEN; SMITH; YORK, 2003) recomenda itens preferidos por usuários que têm afinidade com um determinado *item ativo*. Em sistemas de Filtragem Colaborativa, o acesso do sistema se limita aos identificadores de itens e usuários, sem o uso de informações adicionais. Por exemplo, sites que apresentam recomendações com o título "usuários que preferiram este item também preferem", normalmente usam algum tipo de algoritmo de filtragem colaborativa (GUNAWARDANA; SHANI, 2009). A distinção entre as duas abordagens pode ser melhor observado no esquemático da Figura 1.

Figura 1 – Diferença entre os algoritmos FC baseados em vizinhança



Fonte: Elaborada pelo autor.

Os métodos de filtragem baseados em vizinhança podem ser vistos como uma generalização do algoritmo *k-Nearest Neighbours* (kNN), devido ao uso de medidas de similaridade para fazer predições e recomendações. Com o propósito de avaliar o grupo de um novo usuário, o kNN calcula os K-vizinhos mais próximos a esse usuário e classifica-o como sendo do grupo que aparece com maior frequência dentre os seus K-vizinhos. Algumas medidas utilizadas para calcular a similaridade entre os itens ou usuários são: a correlação de Pearson, cosseno (GUNAWARDANA; SHANI, 2009), diferença média quadrática e entropia (MELVILLE; SINDHWANI, 2010).

2.1.1.2 Baseado em modelo

Em métodos baseados em modelo, um modelo sumarizado dos dados é criado primeiro, assim como acontece com métodos de aprendizagem de máquina. Ao contrário da filtragem baseada em vizinhança, não é usado todo o conjunto de dados para computar previsões reais. Dessa forma, o treinamento (a fase de construção do modelo) é claramente separado da fase da predição. Em consequência, essa técnica tende a produzir recomendações mais rapidamente, depois que os modelos já estão treinados.

Exemplos de técnicas de aprendizado de máquina comumente utilizadas para recomendações incluem métodos de agrupamento, regras de associação, e Redes Neurais (AGGARWAL, 2015). Mais detalhes sobre Redes Neurais serão descritos na Seção 2.2.

Modelos de fatores latentes, como a *Matrix Factorization* (MF), estão entre os métodos mais populares de Filtragem Colaborativa e têm sido o estado da arte em sistemas de recomendação por mais de uma década (BOKDE; GIRASE; MUKHOPADHYAY, 2015), (RENDLE et al., 2020).

Sistemas que usam Filtragem Colaborativa dependem da sobreposição de avaliações (notas) entre usuários e pode ser problemático quando o espaço de avaliações é muito esparso: poucos usuários avaliaram os mesmos itens. O problema da esparsidade é reduzido na abordagem baseada em modelo, com a redução de dimensionabilidade da matriz original de dados, realizada por fatoração de matrizes (MF). Além de fornecer uma abordagem poderosa para lidar com o desafio da escassez de dados, serve para personalizar recomendações com base nos padrões latentes presentes nos comportamentos dos usuários.

Para isso, um dos métodos bastante utilizados é a *Singular Value Decomposition* (SVD) (STRANG, 1988), que decompõe a matriz de avaliações visando descobrir relações latentes entre os usuários e produtos. SVD é o algoritmo de MF mais conhecido (SARWAR et al., 2000), mas existem técnicas mais recentes como *Principal Component Analysis* (PCA) e o *Linear Discriminant Analysis* (LDA) (MÜLLER; GUIDO, 2016).

A *Matrix Factorization* (MF) se destaca por sua capacidade de aprender

representações latentes significativas para usuários e itens, projetando os dados de avaliações em um espaço de menor dimensionalidade[1]. Isso não apenas permite uma representação mais compacta dos padrões de preferência do usuário, mas também facilita o cálculo eficiente de previsões de notas para itens ainda não avaliados. As representações latentes referem-se às características subjacentes não observadas aprendidas pelo modelo para representar usuários e itens. No contexto específico de sistemas de recomendação, essas representações latentes capturam características intrínsecas e não diretamente observáveis dos usuários (como gostos por determinados gêneros de filmes, artistas ou estilos de música) e dos itens (como a complexidade de um livro, a ação em um filme ou a intensidade em uma música) que influenciam as preferências dos usuários (NOGUEIRA et al., 2015).

A concepção por trás desse algoritmo reside na projeção de usuários e itens em um espaço latente compartilhado, utilizando um vetor que encapsula características latentes para representar tanto um usuário quanto um item. Posteriormente, a dinâmica da interação entre um usuário e um item é expressa por meio do produto escalar de seus vetores latentes. (HE et al., 2017).

Tem-se $\mathbf{p}_u$ e $\mathbf{q}_i$ representando os vetores latentes para um usuário u e um item i, respectivamente. A predição da avaliação (nota) y_{ui} é obtida pelo produto interno dos vetores $\mathbf{p}_u$ e $\mathbf{q}_i$, conforme representado pela Equação 2.1.

$$\hat{y}_{ui} = f(u, i \,|\, \mathbf{p}_u, \mathbf{p}_i) = \mathbf{p}_u^T \mathbf{p}_i = \sum_{K=1}^{K} p_{uk} q_{ik}, \tag{2.1}$$

onde K representa a dimensão do espaço latente. O modelo de MF representa a interação recíproca entre os fatores latentes associados ao usuário e ao item. Essa abordagem presume a independência entre cada dimensão do espaço latente, combinando-as linearmente com pesos iguais. Portanto, a fatoração de matrizes é caracterizada como um modelo linear de fatores latentes. O resultado obtido por meio do produto escalar visa ser uma matriz que se assemelha bastante à matriz original de dados, que é, por natureza, esparsa (HE et al., 2017).

[1] A dimensionalidade refere-se ao número de dimensões ou características utilizadas para representar os itens e os usuários em um conjunto de dados.

2.1.1.3 O Problema da Partida a Frio

Um dos maiores problemas que podem ser encontrados ao usar-se FC é quando o número de dados disponíveis sobre as preferências do usuário é inicialmente baixo. Nesses casos, fica mais difícil aplicar modelos que usam filtragem colaborativa tradicional. Esse problema é conhecido como **partida a frio de usuários** (GRAUS; WILLEMSEN, 2015).

De forma similar, quando há itens que não têm avaliações ou que possuem poucas avaliações disponíveis, o sistema de recomendação também executa mal. Esse problema é conhecido como problema de **partida a frio de itens** (NOGUEIRA et al., 2015).

Sistemas híbridos, que combinam abordagens de filtragem colaborativa e baseada em conteúdo, têm o potencial de mitigar os problemas de esparsidade de dados. A informação adicional proveniente dos atributos do item (conteúdo) pode ser valiosa quando associados a filtragem colaborativa. (KO et al., 2022)

2.1.2 Filtragem Baseada em Conteúdo

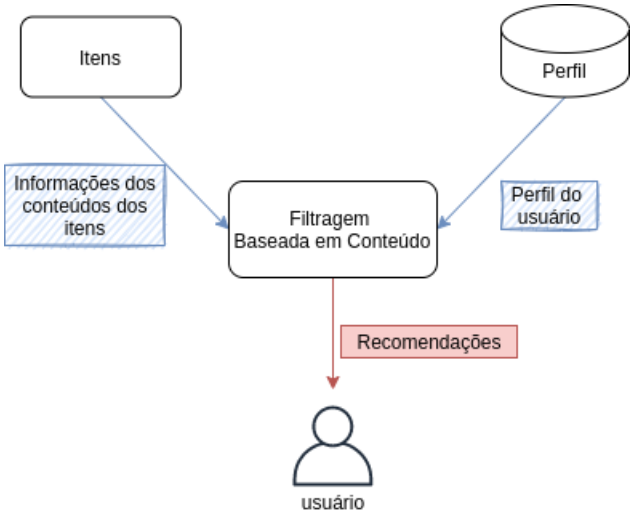
Em sistemas com Filtragem Baseada em Conteúdo (FBC), o usuário receberá recomendações de itens similares àqueles que preferiu no passado. Nessa abordagem, o foco principal está nos atributos dos itens (conteúdo), e as recomendações são feitas com base na similaridade entre esses atributos e as preferências ou histórico do usuário. (AGGARWAL et al., 2016).

Para um FBC funcionar, atributos dos itens que deseja-se recomendar precisam ser extraídos. Atributos sobre o usuário, se disponíveis, tais como características demográficas (*e.g.*, gênero, idade, localização geográfica) também podem fornecer informações valiosas. Dois desafios encontrados em FBCs são (i) uma quantidade de conteúdo limitada para análise e (ii) superespecialização (ADOMAVICIUS; TUZHILIN, 2005). Uma situação que envolve a ausência de conteúdo surge devido à impossibilidade de extrair informações úteis de vários itens em um conjunto de dados, como textos, imagens, áudio, video, etc. Com isso, a qualidade de uma recomendação diminui. Isso também ocorre quando o usuário não avaliou um número suficiente de itens, útil para a criação do *perfil do usuário* - que relaciona

os atributos dos vários itens aos interesses do usuário (notas). Por esse motivo, um usuário novo, com poucas avaliações, não terá recomendações muito precisas.

O problema da superespecialização é causado devido à sugestão de itens muito parecidos com aqueles favorecidos pelo usuário. Isso pode impedir o usuário de receber sugestões inéditas (fora de sua "bolha"), das quais ele poderia gostar (WALEK; FAJMON, 2023). Então, em um nível básico, um sistema de recomendação baseado em conteúdo é dependente de duas fontes de dados: os atributos de vários itens e o *perfil do usuário* (AGGARWAL et al., 2016), como ilustrado na Figura 2.

Figura 2 – Filtragem baseada em conteúdo



Fonte: Adaptado de Nogueira et al. (2015)

2.1.3 Filtragem Híbrida

Sistemas de recomendação híbridos são a integração de duas ou mais categorias de estratégias de recomendação. Um sistema de recomendação híbrido combina os pontos fortes de vários sistemas de recomendação para obter um desempenho mais robusto em uma variedade de aplicações.

Burke (2002) categorizou cinco tipos de sistemas de recomendações por meio de distinções pela sua abordagem. Além dos métodos citados neste capítulo, o autor da pesquisa também categoriza: a filtragem demográfica, a filtragem baseada

em conhecimento e a filtragem baseada em utilidade. A filtragem demográfica usa características demográficas do usuário para identificar preferências em comum com pessoas do mesmo grupo. Os sistemas de recomendação baseados em conhecimento utilizam conhecimento externo de como um item pode atender o usuário para criar recomendações. Sistemas baseados em utilidade aplicam uma função de utilidade sobre os itens que descrevem as preferências de um usuário. Um exemplo de função de utilidade é o lucro que pode ser obtido ao recomendar um item.

Métodos híbridos aumentam a carga do processamento de um sistema, mas devido à disponibilidade de máquinas mais poderosas atualmente, eles são frequentemente usados em sistemas de recomendação para torná-los mais robustos. Algumas estratégias de combinações para gerar as recomendações citadas na literatura (BEZERRA, 2004)(KIM et al., 2010) são:

- Fusão Ponderada: Combinação de diferentes métodos de recomendação com atribuição de pesos. Uma combinação linear é feita com os resultados da filtragem baseada em conteúdo e a filtragem colaborativa.
- Fusão Sequencial: Aplicação sequencial de diferentes métodos de recomendação. A filtragem baseada em conteúdo cria os perfis dos usuários e, posteriormente, estes perfis são usados no cálculo da similaridade na filtragem colaborativa (BALABANOVIC; SHOHAM, 1997).
- Fusão no Modelo: Combinação de características derivadas de diferentes métodos de recomendação, como a junção de características de conteúdos dos items (filtragem por conteúdo) e notas de usuários (filtragem colaborativa) em um mesmo modelo de predição. Tal método pode ser exemplificado por este trabalho.

2.2 Redes Neurais Artificiais

Uma Rede Neural Artificial (RNA) é um conjunto de nós interconectados e links ponderados inspirados na arquitetura do cérebro biológico. Os nós em uma RNA são chamados de neurônios em analogia aos neurônios biológicos. Estas simples

unidades funcionais são compostas em redes com a capacidade de aprender um problema de classificação ou regressão após serem treinadas com dados suficientes (ZURADA, 1992). Cada conexão entre neurônios tem um peso associado, e o aprendizado ocorre ajustando esses pesos com base nos dados. No contexto de sistemas de recomendação, as redes neurais oferecem uma poderosa capacidade de aprendizado de padrões complexos e representações latentes. Uma RNA pode ter várias camadas, e a maior vantagem das redes neurais é que elas podem realizar tarefas não-lineares, e dessa maneira, operar de forma mais eficiente. (GOODFELLOW; BENGIO; COURVILLE, 2016)

Em sistemas de recomendação híbridos, que combinam abordagens colaborativas e baseadas em conteúdo, as redes neurais desempenham o papel de facilitar a incorporação de informações heterogêneas, como avaliações (notas) explícitas, características dos itens e comportamento do usuário, proporcionando uma representação latente abrangente para uma recomendação mais personalizada (MONTI; BRONSTEIN; BRESSON, 2017).

Com o surgimento do *deep learning*, de tradução literal "Aprendizagem Profunda", os últimos anos testemunharam tremendo sucesso em sistemas de recomendação em muitos sites online e aplicações. Por exemplo, um Sistemas de Recomendação (SR) baseado em *Recurrent Neural Network* (RNN) para o Yahoo Notícias ou as *Deep Neural Network* (DNN) em recomendação de vídeos para o YouTube são revoluções significativas nesse campo com a ajuda de *deep learning* (ZHANG et al., 2019)

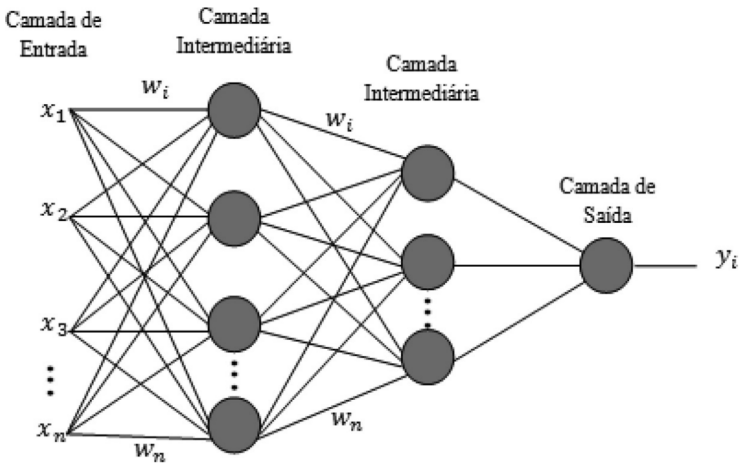
2.2.1 Treinamento de uma Rede Neural Profunda (DNN)

Deep Neural Network (DNN) são modelos compostos por múltiplas camadas de unidades de processamento chamadas neurônios. Cada camada realiza transformações nos dados de entrada, permitindo que o modelo aprenda representações mais abstratas à medida que avança nas camadas.

Uma DNN típica é uma rede totalmente conectada que consiste em uma camada de entrada que recebe dados de entrada, uma camada de saída que toma uma decisão ou faz uma previsão sobre o sinal de entrada, e uma ou mais camadas

ocultas entre essas duas, consideradas como o motor computacional da rede (HAN; PEI; KAMBER, 2011). A saída de uma rede DNN é determinada usando uma variedade de funções de ativação, também conhecidas como funções de transferência, como *rectified-linear-units* (ReLU), Linear, Sigmoide e Softmax (PEDREGOSA et al., 2011). A Figura 3 exibe a estrutura da rede neural adotada no trabalho.

Figura 3 – Ilustração da arquitetura da rede neural artificial adotada.



Fonte: Coutinho et al. (2016)

Para treinar a rede, é empregado o algoritmo mais amplamente utilizado "Backpropagation"(HAN; PEI; KAMBER, 2011), uma técnica de aprendizado supervisionado, que também é conhecida como o bloco básico mais fundamental de uma rede neural. Durante o processo de treinamento, várias abordagens de otimização, como Gradiente Descendente Estocástico (SGD), e Estimação Adaptativa de Momento (Adam), são aplicadas. A DNN requer ajuste de vários hiperparâmetros, como o número de camadas ocultas, neurônios e iterações, o que pode tornar a resolução de um modelo complicado computacionalmente cara (PEDREGOSA et al., 2011).

Alguns outros conceitos relacionados ao treinamento de uma DNN também são necessários para o entendimento deste trabalho.

- Inicialização de pesos: durante o treinamento de uma DNN, os pesos das

conexões entre os neurônios são ajustados iterativamente para minimizar a função de perda.

- Regularização de camadas: 'L1' e 'L2' referem-se a penalidades adicionadas aos pesos da rede para evitar *overfitting*. A regularização L1 adiciona uma penalidade proporcional à soma absoluta dos pesos, enquanto a regularização L2 adiciona uma penalidade proporcional à soma dos quadrados dos pesos.(MÜLLER; GUIDO, 2016)

- *Dropout*: é uma técnica de regularização que envolve aleatoriamente "desligar"(ou "eliminar") uma proporção fixa de neurônios em cada camada durante o treinamento. Isso significa que, em cada iteração do treinamento, um subconjunto aleatório de neurônios é temporariamente removido da rede. (HINTON et al., 2012)

- *Embeddings* são uma estratégia no campo de aprendizado de máquina usada para representar dados de maneira concisa e significativa em um espaço dimensional reduzido. Em sistemas de recomendação, *embeddings* podem ser aplicados para representar usuários e itens (por exemplo, livros) em um espaço latente. Cada usuário e item é vinculado a um vetor de números reais (o embedding), no qual os valores próximos indicam preferências ou características semelhantes. Durante o treinamento do modelo de recomendação, esses *embeddings* são ajustados, permitindo que o sistema capture padrões nas interações entre usuário e item (PROVOST; FAWCETT; BOSCATO, 2016).

2.3 Métricas de Avaliação

De maneira genérica, as avaliações atribuídas a itens são valores numéricos que devem ser estimados. Portanto, as métricas estatísticas para determinar a *acurácia* de uma recomendação são frequentemente as mesmas utilizadas em modelos de regressão. Elas comparam o valor numérico de uma nota prevista com as notas reais entre os pares de usuários-itens armazenadas no *dataset* de teste (SARWAR et al., 2000).

Alternativamente, muitos sistemas de recomendação não preveem as notas; apenas têm como resultado os rankings dos top-k itens recomendados (AGGARWAL et al., 2016). No entanto, neste trabalho serão avaliadas apenas a predição de valores numéricos em uma escala contínua.

2.3.1 Medidas de Acurácia Preditiva

Dentre os métodos que computam a medida de exatidão preditiva, segundo os autores James et al. (2013), encontram-se:

MAE

O Erro Absoluto Médio (MAE) mede a média das diferenças absolutas entre as previsões do modelo e os valores reais, representado pela Equação 2.2. Ele fornece uma indicação do tamanho médio dos erros. Um MAE mais baixo indica que o modelo tem uma tendência a fazer previsões mais precisas, sendo menos sensível a valores extremos.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i| \tag{2.2}$$

MSE

O Erro Quadrático Médio (MSE) penaliza erros grandes mais severamente em comparação com o MAE, já que acrescenta ao erro total o quadrado da diferença entre o valor previsto e o valor real da avaliação (nota). É definido segundo a Equação 2.3:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{2.3}$$

RMSE

A Raiz do Erro Quadrático Médio (RMSE) é a raiz quadrada do MSE. Ele mantém a mesma unidade da variável de resposta original, tornando a interpretação

mais intuitiva. O RMSE é uma métrica comum para avaliar a precisão de um modelo. A fórmula é representada na Equação 2.4:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2} \tag{2.4}$$

Nas equações 2.2, 2.3 e 2.4, temos que:

- n é o número total de observações;
- Y_i é o valor real da observação i;
- $\hat{Y}_i$ é a previsão ou estimativa para a observação.

As medidas MAE, MSE e RMSE são métricas estatísticas de precisão, e quanto mais próximos de zero forem encontrados seus valores, melhor é a acurácia das recomendações.

2.4 Trabalhos Relacionados

Esta seção apresenta uma breve análise de algumas propostas de sistemas de recomendação de livros existentes na literatura recente.

Kiran, Kumar e Bhasker (2020) propõem um sistema de recomendação híbrido para preencher as lacunas existentes em sistemas baseados em filtragem colaborativa e atinge uma precisão de estado da arte na previsão usando *deep learning*. A solução usa *embeddings* para representar usuários e itens, a fim de aprender fatores latentes não lineares, alivia o problema do *Cold Start* ao integrar informações secundárias sobre usuários e itens em uma *Deep Neural Network* (DNN). Experimentos são conduzidos em bases de dados gratuitos de filmes, livros, música, entre outros.

O trabalho de Chandak, Girase e Mukhopadhyay (2015) propõe uma técnica híbrida de otimização de sistemas de recomendação que usa filtragem colaborativa e filtra os usuários que avaliaram os livros recomendados. Uma filtragem posterior é feita baseada em características demográficas. Esses usuários filtrados são então

checados por similaridade com o usuário alvo através da técnica de filtragem por conteúdo para fornecer as recomendações finais.

Kumar e ChawlaLina (2019) propõem um framework para um sistema de recomendação híbrido baseado em *Opinion Mining* (OM). Eles conduzem uma revisão de vários sistemas de recomendação de livros no estado da arte, descrevem e avaliam as técnicas e metodologias mais utilizadas em aplicações que especificadamente recomendam livros e, por fim, é proposto um framework que objetiva tentar resolver lacunas encontradas na literatura revisada, ao implementar uma técnica de recomendação ideal (técnica híbrida), que usa Filtragem Colaborativa, Ontologia, *Sequential Pattern Mining* (SPM) e OM para melhorar o desempenho do sistema.

Os autores em Valdez et al. (2018) focaram em definir um modelo matemático que nos permite criar um algoritmo para converter feedback implícito em feedback explícito em uma plataforma de recomendação de livros eletrônicos. Nove ações implícitas foram avaliadas como positivas por doze algoritmos de aprendizado de máquina populares.

Wang et al. (2018) apresentam um sistema de recomendação de publicações científicas da área de Ciência da Computação baseado em Filtragem Baseada em Conteúdo (FBC). O algoritmo cobre sessenta-e-seis locais de publicação da área em mais de cinco bibliotecas digitais, como Springer, IEEE, ACM, AAAI e SIAM. Não existindo um conjunto de dados para ser usado, foi designado um *web crawler* para coleta de dados e geração de um conjunto de dados para treinamento e testes.

A pesquisa conduzida por Sabitha, Choudhury et al. (2018) concentra-se em prever as avaliações (notas) dos livros feita por novos usuários. Essas avaliações são, então, utilizadas para propor três tipos distintos de recomendações, cada uma baseada em diferentes atributos do usuário. O estudo estabelece uma fundação para o desenvolvimento de sistemas de recomendação, empregando um modelo de previsão de similaridade através do método kNN entre as notas previstas dos usuários, juntamente com a aplicação de filtragem colaborativa.

Sohail, Siddiqui e Ali (2018) propõem um novo sistema de recomendação de livros acadêmicos para ementa de universidades que visa incorporar a opinião de especialistas em conjunto com avaliações (notas) dadas por leitores em sites de venda

online, como a Amazon. É usada uma abordagem de *Opinion Mining* (OM) que explora termos de interpretação de significado recíproco, e realiza uma formulação matemática para conversão de opinião em pontuações.

3 MATERIAIS E MÉTODOS

Neste capítulo são apresentados os materiais e métodos utilizados para o desenvolvimento do trabalho, tais como as ferramentas necessárias, as bibliotecas e *frameworks* em Python, os conjuntos de dados e a modelagem do problema.

3.1 Materiais

Esta seção visa fornecer uma visão geral dos materiais, sendo eles de *software* e/ou *hardware*. Para a implementação dos algoritmos do projeto foi usada a linguagem de programação Python na versão 3.10.12, com o auxílio das bibliotecas TensorFlow, Keras, Pandas, Matplotlib e Seaborn. Durante o desenvolvimento, os experimentos foram conduzidos no ambiente do Google Colab Pro em Jupyter Notebooks, aproveitando os recursos disponíveis na GPU V100. Posteriormente, alguns experimentos foram também conduzidos no Kaggle em uma GPU T4.

3.1.1 Linguagem Python

O motivo da escolha dessa linguagem deve-se, principalmente, a sua ampla popularidade no campo de aprendizado de máquina e *deep learning*. Como resultado, há uma vasta comunidade de desenvolvedores, recursos e bibliotecas disponíveis para suportar projetos de *deep learning*. Python também oferece acesso a bibliotecas de código aberto amplamente utilizadas na área, como Keras e TensorFlow. É uma ferramenta bem documentada e com muitos recursos disponíveis para auxiliar os desenvolvedores na implementação de modelos de aprendizado de máquina.

3.1.2 Bibliotecas e *Frameworks*

TensorFlow e Keras

O TensorFlow[1] é uma biblioteca de código aberto desenvolvida pela Google, amplamente utilizada para construir e treinar modelos de aprendizado profundo. Essa biblioteca oferece uma estrutura flexível para a criação de *Deep Neural Network* (DNN) e outros modelos de aprendizado de máquina.

Os principais componentes do TensorFlow incluem tensores, que são estruturas de dados multidimensionais representando os dados nos modelos, e grafos computacionais, que definem as operações entre esses tensores. A flexibilidade do TensorFlow permite a construção de redes neurais desde as mais simples até arquiteturas avançadas, como *Convolutional Neural Network* (CNN) e *Recurrent Neural Network* (RNN). Além disso, a biblioteca possui interfaces de alto nível, como Keras, que simplificam o desenvolvimento e a experimentação com modelos de aprendizado profundo.

Segundo o site oficial, Keras[2] é uma Interface de Programação de Aplicações (API) funcional para a construção de modelos de *deep learning* escrita em Python, que executa sobre a plataforma TensorFlow. Ela foi desenvolvida com foco em permitir experimentações rápidas, e, portanto, torna-se muito utilizada em estudos científicos. A versão utilizada no desenvolvimento deste trabalho é a versão 2.14.

Pandas

Para manipulação eficiente e análise de conjuntos de dados estruturados, a biblioteca Pandas[3], desenvolvida em Python, oferece estruturas de dados flexíveis e ferramentas para limpeza, manipulação e exploração de dados tabulares. O processo de coleta de dados foi facilitado pela capacidade do Pandas de importar diversos formatos de arquivo, como CSV, Excel e SQL. A limpeza e a transformação dos dados foram simplificadas pela capacidade da biblioteca de lidar com valores ausentes, duplicatas e aplicação de operações em colunas.

[1] https://www.tensorflow.org/overview. Acesso em: 20 dez. 2023.
[2] https://keras.io/. Acesso em: 20 dez. 2023.
[3] https://pandas.pydata.org/. Acesso em: 20 dez. 2023.

Matplotlib e Seaborn

O Matplotlib[4] é uma biblioteca de visualização de dados em Python que oferece uma variedade de ferramentas para criar gráficos estáticos, gráficos interativos e outras representações visuais de dados. Essa biblioteca é amplamente utilizada em contextos científicos, acadêmicos e de desenvolvimento para gerar visualizações de qualidade. Essa biblioteca pode ser empregada para visualizar resultados, como curvas de aprendizado, distribuições de pesos, mapas de ativação ou qualquer outra informação relevante para a análise e interpretação dos modelos implementados.

Seaborn[5] é uma biblioteca para fazer gráficos estatísticos em Python. Ele se baseia no Matplotlib e se integra às estruturas de dados do Pandas. Suas funções de plotagem contém conjuntos de dados inteiros e produzem gráficos informativos. A biblioteca foi utilizada para criar uma melhor visualização dos resultados das métricas no Capítulo 5.

3.1.3 *Datasets*

BookCrossing (BX)

O website BookCrossing[6] permite que uma comunidade de leitores compartilhem seus interesses em livros. Dentro desse sistema, os usuários podem avaliar os livros em uma escala de 1 a 10. O *dataset* completo está disponível no Kaggle [7], contendo 278.858 usuários e 1.149.780 avaliações (explícitas/implícitas) de 271.379 livros.

O *dataset* é composto por 3 tabelas: BX-Users, BX-Books e BX-Book-Ratings. A tabela **BX-Users** contém os IDs (User-ID) dos usuários anonimizados e mapeados para números inteiros. Dados demográficos (Localização, Idade) são fornecidos, se disponíveis. Caso contrário, os campos possuem valores vazios.

Em **BX-Books**, livros são identificados pelos seus respectivos códigos ISBN, sendo que os códigos inválidos já foram removidos do *dataset*. Ademais, algumas

[4] https://matplotlib.org/. Acesso em: 20 dez. 2023.
[5] https://seaborn.pydata.org/. Acesso em: 20 dez. 2023.
[6] https://www.bookcrossing.com/. Acesso em: 20 dez. 2023.
[7] https://www.kaggle.com/datasets/ruchi798/bookcrossing-dataset. Acesso em: 20 dez. 2023.

informações sobre o livro são disponibilizadas (Título do Livro, Autor(a), Ano de Publicação, Editora), obtidos através da Amazon Web Services (AWS). É importante ressaltar que no caso de obras que possuem muitos autores, apenas o primeiro é citado. A versão utilizada neste trabalho conta também com a categoria dos livros.

Na tabela **BX-Book-Ratings** têm-se disponíveis as avaliações (notas) dos livros, dadas pelos usuários. Elas podem ser explicitas, dentro da escala de 1 a 10, ou implícitas, expressas pelo valor 0.

Amazon (AB)

Conforme o trabalho de Ni, Li e McAuley (2019), este *dataset* foi revisado a partir do *dataset* da Amazon[8] disponibilizado em 2014. Semelhante à versão anterior, o conjunto de dados abrange *reviews*, que incluem classificações, conteúdo textual e votos. Ele também incorpora metadados de produtos, consistindo em descrições, detalhes de categoria, preço, informações de marca e recursos de imagem. O conjunto completo compreende um total de 233,1 milhões de avaliações, um aumento notável em relação aos 142,8 milhões de 2014. Os dados abrangem revisões de maio de 1996 a outubro de 2018.

Focando especificamente nas resenhas de livros da Amazon (AB), o *dataset* utilizado no projeto inclui exclusivamente tuplas de (item, usuário, classificação, *timestamp*). As classificações são atribuídas numa escala que varia de 0 a 5, sendo que valores mais elevados indicam maior valorização.

3.2 Métodos

Os problemas que os SR abordam varia dependendo do tipo específico de sistema e da natureza do problema em questão. No entanto, os principais passos ou considerações envolvidos na resolução do problema dos SR são: a coleta de dados; o pré-processamento de dados; a escolha do tipo de SR; o desenvolvimento do modelo; a avaliação do desempenho e a otimização.

[8] https://encr.pw/QtuBO. Acesso em: 20 dez. 2023.

3.2.1 Modelagem do Problema

- Escolha do trabalho de Kiran, Kumar e Bhasker (2020) para investigação e implementação de uma DNN inspirada na arquitetura idealizada pelos autores. O modelo descrito no trabalho funciona de forma híbrida e há a inclusão de *embeddings* com as características dos livros e com a informação de usuários no mesmo modelo de *deeep learning*. A razão para a escolha deste trabalho deve-se a complexidade do modelo e a possibilidade de criar variações e resultados distintos, assim como utilizar uma biblioteca diferente para produzir as recomendações. Os autores utilizam a biblioteca Pytorch[9].

- Obter e pré-processar os *datasets*: os *datasets* foram obtidos através das fontes citadas neste capítulo, e neles foram aplicados métodos de limpeza e tratamento de dados utilizando, na maioria dos processos, a biblioteca Pandas.

- Desenvolvimento e validação do modelo: o desenvolvimento do modelo baseia-se na utilização de dois conjunto de dados para fator de comparação. Apenas o *dataset* BookCrossing (BX) possui dados categóricos adicionais dos livros, sendo assim o *dataset* escolhido para a construção do sistema de recomendação híbrido. O *dataset* da Amazon (AB) é usado para os testes realizados com o modelo de Filtragem Colaborativa (FC) desenvolvido na fase inicial do projeto. A eficácia dos modelos é validada mediante métricas como: MAE,MSE e RMSE.

- Avaliação e ajuste: os dados são divididos em conjuntos de treinamento, validação e teste como procedimento para avaliar a atuação inicial do modelo. Ajustes de hiperparâmetros na arquitetura da DNN para atender às necessidades específicas do projeto também são consideradas. A incorporação ou exclusão de diferentes informações categóricas dos livros utilizadas como entrada do modelo também é ajustado para garantir maior acurácia.

As etapas apresentadas acima são descritas com mais detalhes no Capítulo 4.

[9] https://pytorch.org/. Acesso em: 20 dez. 2023.

4 PROJETO E DESENVOLVIMENTO

Este capítulo aborda as etapas de desenvolvimento do trabalho, dividido em obtenção e processamento dos *datasets*, delimitação do método para a elaboração do modelo, construção das redes neurais e, por último, as etapas dos experimentos realizados.

4.1 Obtenção e pré-processamento dos *Datasets*

Esta seção evidencia as etapas da limpeza e preparação dos dados, destacando os procedimentos específicos adotados para assegurar a integridade e a qualidade dos *datasets* utilizados.

Após a revisão bibliográfica, optou-se por empregar o *dataset* do Book-Crossing, reconhecido por seu uso em estudos acadêmicos recentes, a exemplo de Sabitha, Choudhury et al. (2018) e Kiran, Kumar e Bhasker (2020). Com o intuito de diversificar as avaliações, também foi obtido o conjunto de dados da Amazon, disponibilizado pelos autores Ni, Li e McAuley (2019). Os *datasets* foram obtidos pelo site de seus respectivos disponibilizadores em arquivos compactados, e mais tarde, disponibilizados no Kaggle como contribuição adicional deste trabalho[1].

Uma vez selecionado os dados, o processo de identificação e tratamento de dados ausentes nos *datasets* foi realizado para evitar distorções nos resultados e garantir que a análise fosse realizada com base em dados representativos, assim como as transformações aplicadas nos dados. Na análise feita sobre o *dataset*, a biblioteca Pandas foi utilizada para explorar o conjunto de dados, identificando padrões, estatísticas descritivas e relações entre as variáveis. A abordagem de valores faltantes envolveu estratégias de preenchimento dos valores ausentes ou a remoção, garantindo a qualidade dos dados para análises.

A medição da esparsidade dos dados foi conduzida para compreender a

[1] O *dataset* da Amazon foi disponibilizado no link: https://www.kaggle.com/datasets/brunacmendes/amazon-books-explicit-ratings-2018. Acesso em: 20 dez. 2023.

densidade das informações disponíveis. Esse passo é crucial, pois essa media impacta diretamente a capacidade do modelo de aprender eficazmente.

A separação entre avaliações explícitas (atribuídas diretamente pelos usuários) e avaliações implícitas (dados inferidos de interações, como visualizações ou cliques) foi realizada, para que as avaliações implícitas fossem removidas dos conjuntos de dados. A complexidade de envolver diferentes tipos de avaliações para realizar as predições foi considerada nessa decisão. Os dados também foram divididos entre treinamento e validação, e posteriormente, os dados de treinamento foram divididos novamente para treinar o modelo. Ambas as divisões foram de 90% para treinamento e 10% para validação e teste. Isso foi realizado através do método de validação cruzada (*cross-validation*).[2]

4.2 Delimitação do tema e escolha do método

O escopo do projeto foi definido com base em um problema de modelagem de regressão. O desenvolvimento de um modelo de redes neurais que busca realizar uma previsão de avaliações (notas) dadas por um usuário, produzindo, dessa forma, uma lista de livros para serem recomendados a esse usuário. A decisão por um modelo de DNN é justificada pela inspiração da arquitetura observada no trabalho de Kiran, Kumar e Bhasker (2020), que apresenta um modelo hibrido de *Deep Neural Network* (DNN) que realiza predições numéricas para realizar recomendações de livros. É relevante mencionar que os autores referenciados empregaram a biblioteca *PyTorch* para implementação e desenvolvimento de suas soluções. No entanto, este trabalho optou por utilizar a biblioteca Keras para alcançar os objetivos propostos.

A escolha pelo *framework* Keras decorreu da sua proposta de alto nível para a criação de redes neurais. O uso do Keras visa otimizar o processo de desenvolvimento, permitindo uma concentração mais aprofundada nos aspectos conceituais e metodológicos do sistema de recomendação. A integração nativa do Keras com o TensorFlow proporciona um ambiente mais robusto para a implementação de

[2] A 'validação cruzada' é uma técnica estatística usada em aprendizado de máquina para avaliar o desempenho do modelo. Envolve a divisão do conjunto de dados em subconjuntos para treinamento e teste, garantindo uma avaliação robusta ao evitar dependência de uma única divisão específica dos dados (MURPHY, 2012).

modelos de redes neurais, além de oferecer suporte às últimas inovações e avanços na área. O modelo foi incrementado com informações adicionais sobre os livros e os usuários para criar-se um sistema de recomendação híbrido.

4.3 Arquitetura dos Modelos

Cada usuário e item (livro) é representado por um identificador único (ID). Esses IDs são mapeados para vetores densos (*embeddings*) durante o treinamento do modelo. Uma camada de incorporação (*embeddings* layer) é usada para transformar os IDs de usuários e itens em vetores densos de números reais. As *embeddings* são representações numéricas aprendidas pelo modelo para cada usuário, item ou outra entidade relevante no problema em questão.

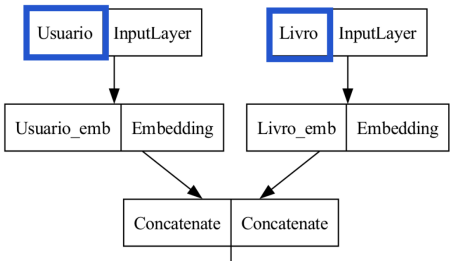
Além das *embeddings*, outras características relevantes dos usuários e itens são incorporadas como variáveis de entrada (X) no modelo híbrido. As representações dos usuários (*embeddings* do usuário) e itens (*embeddings* do livro) são combinadas com as outras características em uma camada de entrada. Isso pode ser feito concatenando os vetores.

Em termos de implementação, isso envolveu o uso de camadas de *embedding* separadas para cada conjunto de características e, em seguida, o concatenamento delas antes de passar para as camadas densas da rede neural. A Figura 4 exibe uma representação visual da entrada das variáveis nos modelos. A principal distinção do modelo híbrido é a introdução das variáveis categóricas dos livros como entradas, conforme destacado na Figura 4b.

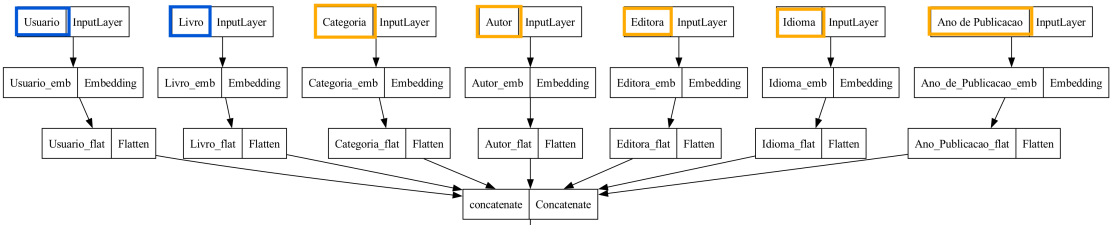
A partir da concatenação, a entrada combinada é passada por várias camadas ocultas da rede neural. Essas camadas aprendem padrões e relações não lineares entre as características de entrada.

A última camada da rede produz a predição final (Y), que representa a estimativa da avaliação (nota) que um usuário daria a um determinado livro.

Uma função de perda é definida para medir a diferença entre as predições do modelo e as avaliações reais. O modelo é treinado usando algoritmos de otimização para minimizar a função de perda, ajustando os parâmetros da rede neural.



(a) Representação da entrada dos vetores de usuários e livros (X) no modelo inicial.



(b) Representação da entrada dos vetores de usuários, livros e informações categóricas dos livros (X) no modelo híbrido.

Figura 4 – Arquitetura da entrada do modelo de DNN.

Após o treinamento, o modelo é avaliado usando dados de validação ou teste para verificar seu desempenho. Para fazer recomendações em tempo real, basta alimentar as informações do usuário e do item no modelo treinado para obter a predição da avaliação.

4.3.1 Utilização de *Features* Adicionais

A inclusão de informações categóricas de usuários e livros serve para melhorar a robustez do modelo e enriquece a análise e o treinamento de modelos. Estas variáveis podem não ser a característica principal do conjunto de dados, mas traz contextos adicionais, detalhes ou relações que podem contribuir para o projeto. Exclusivamente, o conjunto de dados BookCrossing (BX) apresenta estas informações categóricas suplementares sobre os livros, tornando-se, portanto, a opção selecionada para a criação do sistema de recomendação híbrido. Por sua vez, o conjunto de dados da Amazon (AB) é empregado nos testes conduzidos com o modelo sem o uso de *features* adicionais, desenvolvido nas etapas iniciais do projeto.

As informações obtidas para o propósito desta implementação estão descritas na Tabela 1.

Dataset	Informação Adicional	
Bookcrossing	Item	Autor(a), editora, ano de publicação, idioma, categoria
	Usuário	Idade

Tabela 1 – Informações adicionais do *dataset* da BookCrossing (BX)

Ao incluir essas *features* no treinamento, o modelo de recomendação torna-se mais robusto em situações de "*cold start*". Se um usuário ou item não possui um histórico de avaliações (notas), o modelo pode confiar nas informações fornecidas por essas *features* adicionais para fazer recomendações iniciais. Isso permite que o modelo considere tanto as informações latentes aprendidas nas *embeddings* quanto as informações específicas das características ao fazer previsões.

4.4 Construção das Redes Neurais

A estratégia usada para a construção dos modelos utilizando o Keras foi conduzida a partir de modelos-base que foram progressivamente se adaptando ao problema e às variáveis de entrada. Na arquitetura da rede neural, optou-se por utilizar o modo funcional ('Model') do Keras em vez do modelo sequencial ('Sequential'). O modo funcional oferece maior flexibilidade ao permitir a criação de variações mais complexas e com múltiplas entradas. O modo 'Sequential' adiciona camadas sequencialmente, uma após a outra, e não atende aos requisitos específicos do projeto.

Abstraindo-se os detalhes, a construção do modelo teve como seu primeiro passo a criação dos *Embeddings* (linhas 2 a 7 do Quadro 1), que constituem os vetores densos que capturam características importantes de cada categoria. O parâmetro *output_dim* é a dimensão do *embedding*, *input_dim* é o número total de itens, e *input_length* é o comprimento da sequência de entrada. É necessário que essa etapa seja realizada com todos os atributos que serão usados na entrada do modelo, mesmo que no Quadro 1 isso tenha sido exemplificado apenas com o

Input do vetor **item**, representado neste cenário pelo ID dos livros. Os *embeddings* resultantes são achatados *(Flatten)* na linha 10 e, em seguida, concatenados em um único vetor na linha 13. Isso combina todas as informações relevantes das diferentes informações de entrada em uma representação conjunta. A representação concatenada é alimentada em camadas totalmente conectadas (dense) com ativações ReLU. Cada camada é seguida por uma camada de *Dropout* (linhas 17,21 e 25) para evitar *overfitting*. As regularizações L1 ou L2 também são aplicadas para penalizar coeficientes excessivos. A última camada gera uma única saída, que representa a nota do livro predita, e normalmente se utiliza uma ativação linear (ou nenhuma ativação), especialmente quando se trata de avaliações contínuas. O uso de *activation=None* (linha 31) significa que não há ativação aplicada, e a saída seria um valor contínua. O modelo é compilado com um otimizador SGD e a função de perda é o Erro Quadrático Médio (MSE).

4.4.1 Nomenclatura dos modelos

Optou-se pela criação de uma nomenclatura para cada modelo para facilitar a distinção entre os resultados obtidos pelos experimentos, visto a presença de diferentes variáveis associadas aos dados de entrada. Os modelos foram batizados de dnnCF, dnnBX-H6 e dnnBX-H7.

- **Modelo dnnCF:** A arquitetura é distintiva por utilizar exclusivamente usuários e livros como vetores de entrada. O propósito é discernir os resultados obtidos a partir de um modelo não-híbrido, e este mesmo modelo foi utilizado em ambos os *datasets*.
- **Modelo dnnBX-H6:** Possui uma arquitetura que tem como entrada quatro atributos categóricos (autor, categoria, editora e idioma) além dos IDs de usuários e livros, totalizando seis variáveis de entrada.
- **Modelo dnnBX-H7:** Apresenta essencialmente a mesma arquitetura descrita no modelo dnnBX-H6, mas inclui-se a variável categórica 'ano de publicação' na entrada do modelo para criar-se a oportunidade de comparação entre instâncias onde se varia a quantidade de *embedddings* categóricas. Esse modelo possui sete variáveis de entrada.

Quadro 1 – Código-fonte com o modelo DNN no Keras

```

#criar embeddings
item_input = Input(shape=(1,), name='item_input')
item_embedding = Embedding(output_dim=embedding_size,
                           input_dim=n_books,
                           input_length=1,
                           name='book_embedding')(item_input)

# Flatten the embeddings
item_flat = Flatten()(item_embedding)

# Concatenate all embeddings
merged = concatenate([user_flat, item_flat, genre_flat,
   author_flat,publisher_flat,language_flat,year_flat])

#hidden layer 1
fc1 = Dense(400, activation='relu',
   kernel_regularizer=l1(0.00001))(merged)
fc1_dropout = Dropout(0.2)(fc1)

#hidden layer 2
fc2 = Dense(256, activation='relu',
   kernel_regularizer=l1(0.00001))(fc1_dropout)
fc2_dropout = Dropout(0.1)(fc2)

#hidden layer 3
fc3 = Dense(128, activation='relu')(fc2_dropout)
fc3_dropout = Dropout(0.1)(fc3)

#hidden layer 4
fc4 = Dense(64, activation='relu')(fc3_dropout)

# output layer
output = Dense(1,activation=None)(fc4)

model = Model(inputs=[user_input, item_input, genre_input,
   author_input,publisher_input,language_input,year_input],
   outputs=output)

model.compile(optimizer=SGD(learning_rate=0.001),
   loss='mean_squared_error',
              metrics=['mean_squared_error',
                       'mean_absolute_error',
                       RootMeanSquaredError()])
```

4.5 Etapas do Experimento

4.5.1 Fase I - Variação do Modelo Inicial

A primeira etapa consistiu na avaliação do modelo inicial, cuja entrada tem como valores apenas o IBSN dos Livros e as IDs dos usuários. Alguns testes foram realizados para a avaliação da função de perda (*loss*) do modelo utilizando MSE,MAE e RMSE como métrica. As variações consistiram em alternar os otimizadores entre SGD e ADAM, assim como o número de camadas ocultas da DNN e mudanças de hiperparâmetros, como: (i) regularizadores L1 e L2, *dropout*, e função de ativação. O modelo inicial pode ser considerado ideal para Filtragem Colaborativa, levando em consideração que as variáveis de entrada consistem nos identificadores de usuários e livros e nas avaliações (notas) explicitas. Avaliou-se, principalmente, a possibilidade do treinamento do modelo causar *overfitting* e o resultado das métricas. Os testes foram realizados em ambos os *datasets*. O desempenho da rede também foi levado em consideração devido à variação de camadas para o treinamento.

4.5.2 Etapa II - Variação do Modelo Híbrido

Na segunda etapa do desenvolvimento, realizou-se uma variação do modelo adotando uma abordagem híbrida. Nessa configuração, foram incorporadas novas fontes de informação em forma de *embeddings* para aprimorar a predição de resultados. Além dos usuários e livros, outros elementos foram integrados como vetores de entrada. O objetivo principal é capturar padrões mais complexos e nuances nas interações entre usuários e itens, resultando em previsões mais refinadas. Durante essa etapa, ajustes e otimizações específicos para o contexto da abordagem híbrida foram realizados, visando a maximização do desempenho do modelo. Experimentos foram conduzidos para avaliar a eficácia da inclusão de novas fontes de informação, permitindo uma análise comparativa entre o modelo híbrido e o modelo não-híbrido desenvolvido na etapa anterior.

5 RESULTADOS E ANÁLISE

Neste capítulo, serão apresentados os resultados obtidos durante a implementação e avaliação do sistema de recomendação proposto. Os experimentos foram conduzidos em dois conjuntos de dados distintos para garantir a robustez e generalização do modelo. Esta seção é subdividida em partes que incluem a caracterização dos conjuntos de dados e os resultados alcançados pelos modelos implementados.

5.1 Caracterização dos *Datasets*

A Tabela 2 apresenta informações abrangentes sobre os *datasets*, incluindo a esparsidade dos dados, expressa como o número de avaliações em relação ao total da matriz de usuários e itens. Adicionalmente, a tabela fornece detalhes sobre o número de usuários, o número de livros, a escala das avaliações e o número total de avaliações em cada conjunto de dados.

Tabela 2 – *Datasets* usados para experimentação

Dataset	Usuários	Livros	Avaliações	Esparsidade	Escala
Bookcrossing (BX)	68.092	149.842	383.852	99%	[0-10]
Amazon (AB)	14.741.679	2.270.829	49.574.702	99%	[0-5]

Fonte: Elaborada pelo autor.

Ambos os *datasets* possuem a esparsidade alta de mais de 99%. No conjunto de dados do BookCrossing (BX), foram observadas lacunas nos dados referentes ao idioma, categoria e idade. No contexto da idade, optou-se por aplicar a média correspondente para preencher os valores faltantes. Entretanto, em relação ao idioma e categoria, uma decisão foi tomada para remover parte dos dados que se apresentavam incompletos.

A indexação dos atributos categóricos para números inteiros envolveu a utilização de técnicas como *Label Encoding*. Esse processo facilitou a entrada desses

atributos no modelo, mantendo a integridade semântica dos dados. Para o propósito desde trabalho, as avaliações implícitas foram desconsideradas, pois sua inclusão se demonstra uma tarefa incompatível com o prazo disponível.

No conjunto de dados do BookCrossing (BX), as avaliações originalmente estavam em uma escala de 1 a 10. Para garantir consistência e comparabilidade com outro conjunto de dados, foi necessário normalizar essa escala para variar de 0.5 a 5. Isso foi feito para que ambas as fontes de dados compartilhassem a mesma escala de avaliações, facilitando a integração e a análise conjunta. A normalização permite que as informações de avaliação sejam interpretadas de maneira uniforme, garantindo coerência nos resultados obtidos a partir de diferentes conjuntos de dados.

A Figura 5 representa a dispersão das avaliações contidas nos dois *datasets* distintos.

Por fim, devido ao tamanho substancial do conjunto de dados da Amazon (AB), foi necessário extrair uma amostra aleatória para treinar a rede neural, mantendo uma proporção de 0.006 (0.6%). Essa abordagem preservou a dispersão de avaliações explícitas nos dados.

Tabela 3 – *Datasets* resultantes após pré-processamento de dados

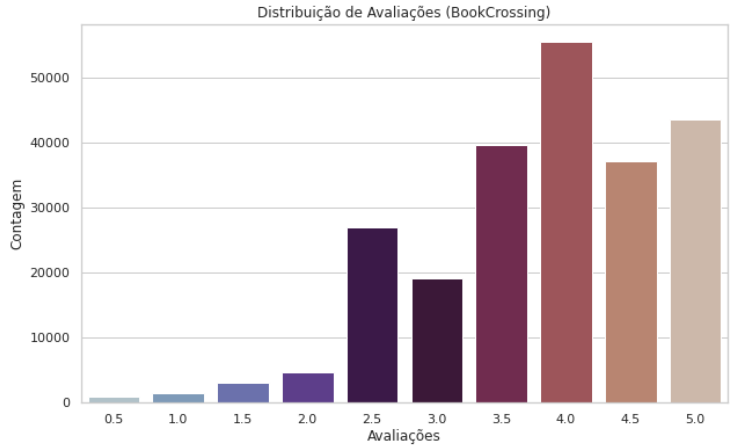
Dataset	Usuários	Livros	Avaliações	Esparsidade	Escala
Bookcrossing (BX)	49.363	80.852	231.585	99%	[0.5-5]
Amazon (AB)	279.498	186.050	307.870	99%	[1-5]

Fonte: Elaborada pelo autor.

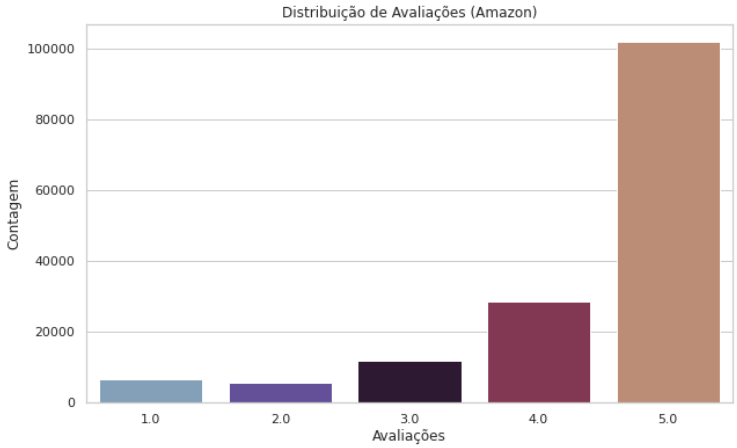
5.2 Resultados Parciais

Na fase de experimentação, foram explorados diversos ajustes na arquitetura da rede neural para otimizar o desempenho do modelo. Inicialmente, observou-se que um número reduzido de camadas estava propenso a causar *overfitting*. Para lidar com esse desafio, optou-se por aumentar o número de neurônios nas camadas.

Foram conduzidos experimentos variando o tamanho do lote *(batch size)* e o número de épocas. Inicialmente, valores maiores foram considerados; no entanto,



(a) Dispersão de avaliações no *dataset* do BookCrossing



(b) Dispersão de avaliações no *dataset* da Amazon

Figura 5 – Distribuição das avaliações de livros na escala [0.5-5]

percebeu-se que um tamanho de lote de aproximadamente 64 e um número mais ampliado de épocas resultavam em desempenho mais satisfatório. Os experimentos foram realizados com 50 a 100 épocas cada.

Nos experimentos iniciais percebeu-se que a variável de perda dos dados de treinamento estavam muito mais altos que a perda dos dados de teste, o que podia indicar que havia mais de um problema com o modelo ou com os dados. O ajustes realizados para reduzir esse problema foi diminuir a taxa de aprendizado

da otimização (para 0.001) e usar uma inicialização de pesos nas camadas. A biblioteca Keras oferece algumas opções, mas os testes ficaram entre 'he_normal' e 'he_uniform', inicializadores ajustados para funcionar melhor com funções de ativação ReLU. No final, optou-se por 'he_uniform'. Para o mesmo propósito, também foi adotada uma estratégia que envolve o uso de uma função capaz de ajustar dinamicamente a taxa de aprendizado durante a otimização, caso a perda (*loss*) não demonstre melhorias por pelo menos duas épocas consecutivas. Essa função é denominada *ReduceLROnPlateau*. A função *ReduceLROnPlateau* monitora o desempenho do modelo ao longo das épocas e reduz a taxa de aprendizado se a melhoria na perda estagnar, contribuindo para um treinamento mais eficaz.

Uma mudança relevante nas experimentações foi a transição do otimizador de Adam para o otimizador SGD. Essa alteração foi motivada pelas propriedades específicas do SGD, que é mais simples em comparação com o Adam. Essa escolha visa avaliar o impacto na convergência do modelo e nos resultados obtidos.

5.3 Resultados Finais

As considerações e decisões que levaram à seleção dos parâmetros selecionados foram as seguintes:

- Para a função de perda (*loss*), uma vez que não se trata de um problema de classificação, a função de perda[1] priorizada foi a Erro Quadrático Médio (MSE);
- A escolha do otimizador é feita ao perceber que o SGD se apresenta como um dos melhores, sendo menos sensível a escolhas específicas de hiperparâmetros em comparação ao Adam, o que pode simplificar o ajuste do modelo;
- O *dropout* é utilizado na primeira camada densa com o valor [0.2].
- Finalmente, a seleção da função de ativação foi fundamentada nos resultados obtidos a partir de experimentos conduzidos durante a avaliação de outras

[1] Durante o treinamento de uma DNN, os pesos das conexões entre os neurônios são ajustados iterativamente para minimizar a função de perda

funções. Contudo, constatou-se que os índices de perda e RMSE foram menos favoráveis ao empregar essas alternativas em comparação com a utilização de ReLU para as camadas de espaço latente, e nenhuma para a camada de saída.

Função de Perda (Loss)	Erro Quadrático Médio (MSE)
Otimizador	SGD com *learning rate* inicial=0.001
Dropout	0.2
Função de ativação	ReLU para espaço latente, nenhuma na *output* layer
Dimensão embeddings (output_dim)	50
Camadas Profundas	[400,128,64,10]

Tabela 4 – Características dos modelos.

5.3.1 Revisão Comparativa entre Diferentes Modelos

A Figura 6 mostra o desempenho do modelo dnnCF usando o conjunto de dados da Amazon (AB). Em termos de perda no treinamento *(loss)* e perda durante a validação, elas diminuem a cada época, com algumas situações pontuais onde aumentam antes de serem ajustadas dinamicamente, e atingem um ponto onde começam a se estabilizar.

Não há interseção entre a curva de perda no treinamento e na validação porque, ao adicionar as camadas de *dropout*, elas atuam no conjunto de dados de treinamento e isso resulta na diferença observada entre elas. Se isso não tivesse sido adicionado, o resultado seria um modelo com *overfitting*.

A Figura 7 apresenta a perda do modelo para o conjunto de dados BX. Ao analisar a figura, podemos observar um comportamento semelhante ao da Figura 6, mas as perdas apresentam um valor mais baixo e mais linear.

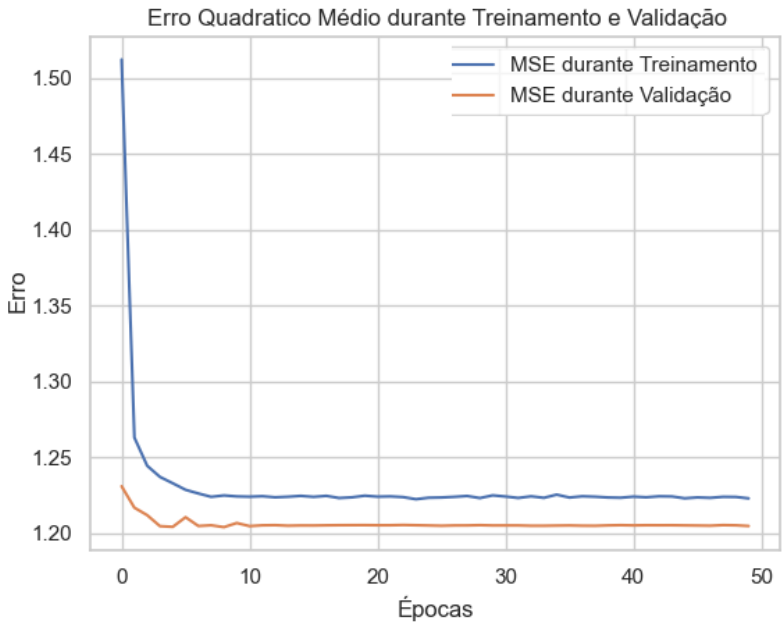


Figura 6 – Erro Quadrático Médio (Função de Perda) no Modelo dnnCF do *dataset* AB.

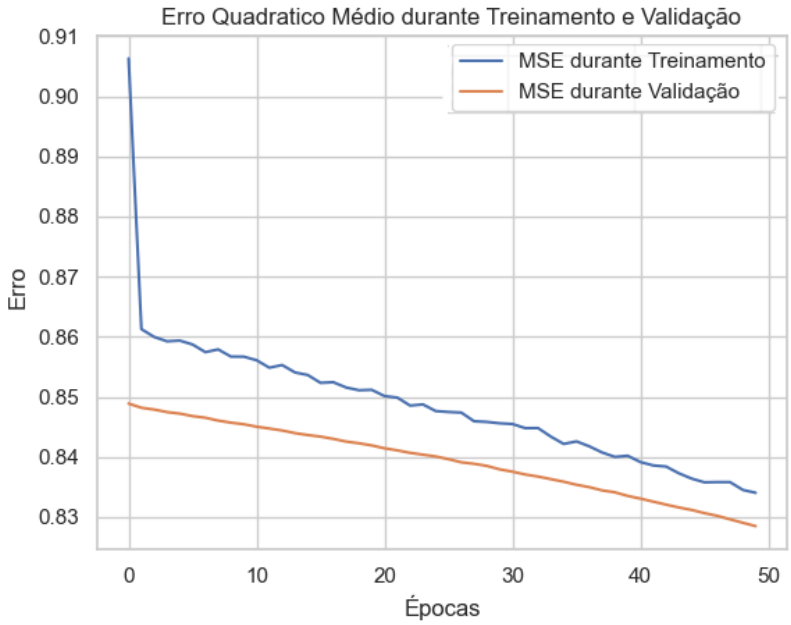


Figura 7 – Erro Quadrático Médio (Função de Perda) no Modelo dnnCF do *dataset* BX.

O gráfico Figura 8 demonstra uma perda de Erro Quadrático Médio (MSE)[2] inicialmente mais aguda que nos modelos iniciais, mas ainda mantendo a tendência de atingir a estabilização após uma quantidade de épocas. Ao contrário da curva observada na Figura 6, a intercessão dos conjuntos de dados de treinamento e teste acontece no início, e isso também pode ser observado no gráfico da Figura 9. A estabilização do treinamento é mais demorado nos modelos híbridos, mas a eficácia observada é maior. A complexidade dos modelos híbridos, especialmente quando incorporam múltiplas entradas, é um fator que contribui para a extensão do tempo de treinamento. A presença de várias entradas resulta em um aumento na dimensionalidade dos dados, demandando mais iterações durante o processo de treinamento para realizar os ajustes necessários nos parâmetros de forma adequada.

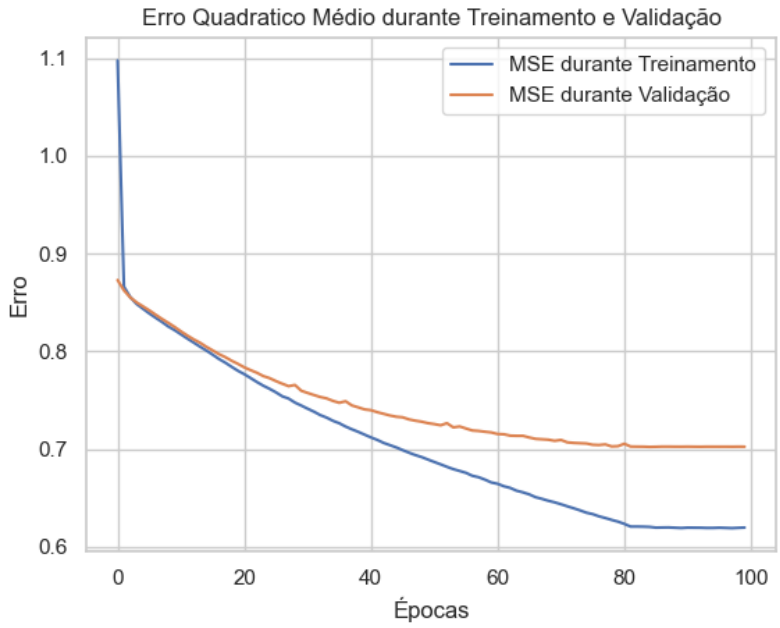


Figura 8 – Erro Quadrático Médio (Função de Perda) no Modelo dnnBX-H6.

[2] MSE é um método que computa a medida de exatidão preditiva, confirme visto na Subseção 2.3.1

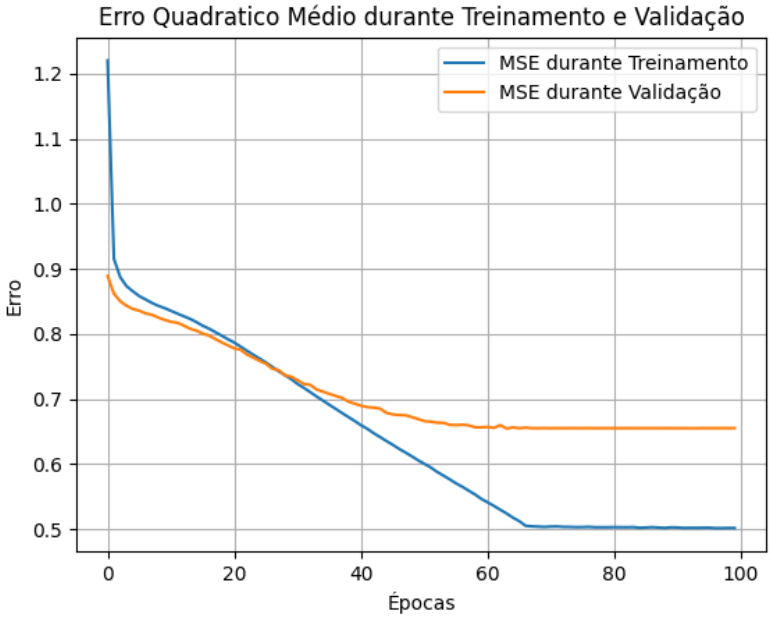


Figura 9 – Erro Quadrático Médio (Função de Perda) no Modelo dnnBX-H7.

Na análise das curvas de treinamento, nota-se um fenômeno ocorrido nas épocas 80 e 70, como evidenciado nas Figuras 8 e 9, respectivamente. Nessas fases específicas do treinamento, observa-se a formação de um platô, indicando uma estabilização nas métricas de desempenho. Isso significa que o modelo alcançou uma fase em que as métricas praticamente não apresentam variações ao longo de várias épocas subsequentes. Tal observação sugere que o modelo pode ter atingido um estágio de estabilidade, indicando uma possível convergência para uma solução ótima no processo de treinamento.

5.3.2 Avaliação da Solução

Na tabela Tabela 5 é possível ver os resultados obtidos, bem como as outras métricas utilizadas para avaliar os experimentos.

Ao analisarmos os resultados apresentados na tabela de comparação entre diferentes métodos aplicados a diversos *datasets*, podemos observar variações significativas nas métricas de erro, evidenciando o desempenho relativo entre eles.

	Dataset	Método	MSE	MAE	RMSE
1	BX	dnnCF	0.8285	0.7373	0.9102
2	AB	dnnCF	1.0934	0.8247	1.0457
3	BX	dnnBX-H6	0.6978	0.6505	0.8353
4	BX	dnnBX-H7	0.6554	0.6253	0.8096

Tabela 5 – Comparação dos métodos nos diferentes *datasets*. Cores indicam melhor exatidão preditiva em cada métrica.

Em relação ao método dnnCF no *Dataset* BX, notamos melhorias[3] em várias métricas ao considerar os métodos alternativos. O método dnnBX-H6, por exemplo, demonstra uma melhoria de aproximadamente 18,73% no MSE, indicando uma redução significativa no erro quadrático médio em comparação com o dnnCF. Similarmente, o método dnnBX-H7 apresenta uma melhoria de cerca de 26,41% no MSE, sugerindo um desempenho superior na minimização do erro quadrático médio.

No que diz respeito ao RMSE, ambas as variantes dnnBX-H6 e dnnBX-H7 superam o dnnCF do *dataset* BX, com melhorias de cerca de 8,96% e 12,42%, respectivamente. Contrastando esses resultados, o dnnCF no Dataset AB revela uma elevação de aproximadamente 14,88% no RMSE, sugerindo uma adaptação menos eficaz a esse conjunto de dados específico.

A comparação entre os resultados nos *datasets* BX e AB destaca a importância de considerar as características únicas de cada conjunto de dados. Métodos que se saem bem em um *dataset* podem não apresentar o mesmo desempenho em outro, evidenciando a necessidade de ajustes específicos para cada contexto.

A diferença nos resultados entre as configurações dnnBX-H6 e dnnBX-H7 no mesmo *dataset* BX foi pequena, mas ainda abre precedentes para a possibilidade de que alterações na configuração de entrada de dados influenciarem no resultado.

[3] A melhoria percentual é calculada pela fórmula: Melhoria $= \frac{\text{Métrica\_Método1} - \text{Métrica\_Método2}}{\text{Métrica\_Método2}} \times 100$. Onde: Métrica_Método1 é o valor da métrica para o Método 1 (por exemplo, MSE para dnnCF) e Métrica_Método2 é o valor da métrica para o Método 2 (por exemplo, MSE para dnnBX-H6).

5.3.3 Validação das Predições

A seguir, são apresentadas as cinco principais avaliações geradas pelo modelo dnnBX-H7 para um usuário específico durante o processo de validação. A tabela abaixo exibe as avaliações reais e as respectivas avaliações obtidas:

Livro	**Avaliação Real**	**Previsão do Modelo**
The Color Purple	4.5	4.2974
The Chosen	4.5	4.1982
Native American Songs and Poems: An Anthology	4.0	4.0538
Davita's Harp	3.5	4.0449
Oliver Twist	3.5	3.8215

Tabela 6 – Top 5 Avaliações de um usuário específico no conjunto de dados BX.

A Tabela 6 acima revela a eficácia do modelo ao prever as avaliações de filmes para o usuário em questão. As previsões do modelo são próximas às avaliações reais, indicando um desempenho confiável.

Vale ressaltar que, durante o processo de validação, a previsão das avaliações do usuário específico foi realizada em 0.8 segundos, tornando-o uma escolha viável para aplicações em tempo real.

5.3.4 Limitações da Pesquisa

Como limitação da pesquisa, é importante destacar que não foi possível realizar uma comparação com os resultados de outros trabalhos devido à grande variedade de modelos encontrados na literatura, tornando desafiador oferecer uma comparação relevante. Outra limitação decorreu da escassez de tempo, impedindo a utilização de uma busca em grade (*grid search*) para aprimorar os parâmetros do modelo; todos os testes foram conduzidos manualmente. Essa limitação pode ter impactado os resultados, uma vez que uma busca em grade teria permitido uma combinação mais otimizada de parâmetros, elevando potencialmente a acurácia. Essa lacuna pode ser considerada para trabalhos futuros como uma extensão deste estudo. A limitação de memória RAM também foi um desafio durante o treinamento do modelo, impedindo a expansão para conjuntos de dados mais extensos.

Em conclusão, este trabalho proporcionou *insights* valiosos em relação ao desempenho de diferentes modelos de redes neurais. Espera-se que as informações apresentadas neste estudo sejam úteis para área de Sistemas de Recomendação. A pesquisa é um processo contínuo, e as descobertas deste estudo podem ser bases para investigações futuras nesta área.

6 CONSIDERAÇÕES FINAIS

Neste capítulo, o objetivo é consolidar os *insights* adquiridos ao longo deste trabalho e ressaltar a importância das descobertas, assim como delinear possíveis caminhos para avanços posteriores na área de sistemas de recomendação.

6.1 Conclusão

Uma das principais considerações deste estudo é a evidência de que sistemas de recomendação híbridos demonstram uma capacidade para realizar previsões eficazes e apresenta uma melhora em comparação a sistemas tradicionais. A complexidade inerente aos modelos híbridos, que combinam abordagens colaborativas e baseadas em conteúdo, podem ser interessantes para aprimorar a precisão e personalização das recomendações.

A constatação de que a incorporação de elementos híbridos melhoram as previsões destaca a necessidade contínua de inovação nesse campo. A capacidade de adaptar modelos às características específicas dos dados e do domínio torna-se evidente como um ponto para avanços futuros. No contexto dos sistemas de *e-commerce* de livros, a implementação de sistemas de recomendação eficazes pode não apenas otimizar as sugestões dos produtos, mas também promover a fidelidade do cliente.

Sistemas de recomendação abrem novas oportunidades para fornecer informações personalizadas na internet, e também ajuda a aliviar o problema da sobrecarga e expansão temporal de dados online.

6.2 Trabalhos Futuros

Como ponto de partida para pesquisas futuras, podem ser explorados determinados aspectos, tais como:

- Desenvolver uma interface para a realização das recomendações.
- Adaptação do modelo para conjunto de dados maiores que exigem maior poder de processamento, para fins de escalabilidade.
- O impacto do crescimento dos dados ao longo do tempo no desempenho dos sistemas de recomendação.
- O uso de diferentes algoritmos para otimização de hiperparâmetros no modelo de DNN como: otimização Bayesiana ou *grid search*.
- Fazer o uso de modelos de DNN diferentes para comparação de desempenho, como Autoencoders.
- A medição do desempenho dos modelos através do cálculo e comparação do tempo de treinamento.
- Ampliar a aplicação deste modelo híbrido em um ambiente prático, envolvendo usuários reais, mediante a implementação de um sistema que inclui um webservice e uma API para solicitar recomendações.
- Estudo da possibilidade de generalização para realização de predições a partir de bases de dados de outros tipos de mídia, como músicas ou filmes.

Este trabalho contribui para o entendimento da dinâmica dos sistemas de recomendação no contexto específico de recomendação de livros, e espera-se que estimule novas pesquisas e avanços neste campo em constante evolução. Além disso, destaca-se pela proposição de um modelo cuja aplicação pode ser replicada em diferentes contextos e estudos.

Referências

ADOMAVICIUS, G.; TUZHILIN, A. Extending recommender systems: A multidimensional approach. In: CITESEER. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-01), Workshop on Intelligent Techniques for Web Personalization (ITWP2001), Seattle, Washington, August*. [S.l.], 2001. p. 4–6. Citado na página 15.

ADOMAVICIUS, G.; TUZHILIN, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, IEEE, v. 17, n. 6, p. 734–749, 2005. Citado 2 vezes nas páginas 19 e 24.

AGGARWAL, C. C. *Data mining: the textbook*. [S.l.]: Springer, 2015. Citado na página 22.

AGGARWAL, C. C. et al. *Recommender systems*. [S.l.]: Springer, 2016. v. 1. Citado 5 vezes nas páginas 19, 20, 24, 25 e 30.

BALABANOVIC, M.; SHOHAM, Y. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, v. 40, p. 66–72, 03 1997. Citado na página 26.

BEZERRA, B. L. D. *Uma solução em filtragem de informação para sistemas de recomendação baseada em análise de dados simbólicos*. 108 p. Dissertação (Dissertação) — Centro de Informática, Universidade Federal de Pernambuco, 2004. Pós-Graduação em Ciência da Computação. Citado na página 26.

BOKDE, D. kumar; GIRASE, S.; MUKHOPADHYAY, D. Role of matrix factorization model in collaborative filtering algorithm: A survey. *CoRR, abs/1503.07475*, 2015. Citado na página 22.

BURKE, R. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, Springer, v. 12, n. 4, p. 331–370, 2002. Citado 2 vezes nas páginas 19 e 25.

CHANDAK, M.; GIRASE, S.; MUKHOPADHYAY, D. Introducing hybrid technique for optimization of book recommender system. *Procedia Computer Science*, Elsevier, v. 45, p. 23–31, 12 2015. Citado na página 31.

COUTINHO, E. et al. Using computational intelligence technique for the meteorological data prediction. *Revista Brasileira de Meteorologia*, v. 31, p. 24–36, 01 2016. Citado na página 28.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. MIT Press, 2016. (Adaptive Computation and Machine Learning series). ISBN 9780262035613. Disponível em: <https://books.google.com.br/books?id=Np9SDQAAQBAJ>. Citado na página 27.

GRAUS, M. P.; WILLEMSEN, M. C. Improving the user experience during cold start through choice-based preference elicitation. In: *Proceedings of the 9th ACM Conference on Recommender Systems*. [S.l.: s.n.], 2015. p. 273–276. Citado na página 24.

GUNAWARDANA, A.; SHANI, G. A survey of accuracy evaluation metrics of recommendation tasks. *Journal of Machine Learning Research*, v. 10, n. 12, 2009. Citado na página 21.

HAN, J.; PEI, J.; KAMBER, M. *Data Mining: Concepts and Techniques*. Amsterdam: Elsevier, 2011. Citado na página 28.

HE, X. et al. Neural collaborative filtering. *CoRR*, abs/1708.05031, 2017. Disponível em: <http://arxiv.org/abs/1708.05031>. Citado na página 23.

HINTON, G. E. et al. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012. Citado na página 29.

JAMES, G. et al. *An Introduction to Statistical Learning: with Applications in R*. Springer New York, 2013. (Springer Texts in Statistics). ISBN 9781461471387. Disponível em: <https://books.google.com.br/books?id=qcI_AAAAQBAJ>. Citado na página 30.

KIM, H. N. et al. Collaborative filtering based on collaborative tagging for enhancing the quality of recommendation. *Electronic Commerce Research and Applications*, v. 9, n. 1, p. 73–83, 2010. Citado na página 26.

KIRAN, R.; KUMAR, P.; BHASKER, B. Dnnrec: A novel deep learning based hybrid recommender system. *Expert Systems with Applications*, Elsevier, v. 144, p. 113054, 2020. Citado 4 vezes nas páginas 31, 38, 39 e 40.

KO, H. et al. A survey of recommendation systems: Recommendation models, techniques, and application fields. *Electronics*, v. 11, n. 1, 2022. ISSN 2079-9292. Disponível em: <https://www.mdpi.com/2079-9292/11/1/141>. Citado na página 24.

KUMAR, A.; CHAWLALINA, S. Framework for hybrid book recommender system based on opinion mining. *International Journal of Recent Technology and Engineering (IJRTE)*, v. 08, 09 2019. Citado na página 32.

KUROIWA, T.; BHALLA, S. Dynamic personalization for book recommendation system using web services and virtual library enhancements. In: IEEE. *7th IEEE International Conference on Computer and Information Technology (CIT 2007)*. [S.l.], 2007. p. 212–217. Citado na página 16.

LINDEN, G.; SMITH, B.; YORK, J. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, IEEE, v. 7, n. 1, p. 76–80, 2003. Citado 2 vezes nas páginas 15 e 21.

MELVILLE, P.; SINDHWANI, V. Recommender systems. *Encyclopedia of machine learning*, v. 1, p. 829–838, 2010. Citado na página 21.

MONTI, F.; BRONSTEIN, M.; BRESSON, X. Geometric matrix completion with recurrent multi-graph neural networks. In: GUYON, I. et al. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. v. 30. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2017/file/2eace51d8f796d04991c831a07059758-Paper.pdf>. Citado na página 27.

MÜLLER, A.; GUIDO, S. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media, Incorporated, 2016. ISBN 9781449369408. Disponível em: <https://books.google.com.br/books?id=q5pnAQAACAAJ>. Citado 2 vezes nas páginas 22 e 29.

MURPHY, K. P. *Machine Learning: A Probabilistic Perspective*. [S.l.]: MIT Press, 2012. Capítulo 5 - Cross-Validation p. Citado na página 40.

NI, J.; LI, J.; MCAULEY, J. Justifying recommendations using distantly-labeled reviews and fined-grained aspects. *Empirical Methods in Natural Language Processing (EMNLP)*, 2019. Citado 2 vezes nas páginas 37 e 39.

NOGUEIRA, E. A. et al. Uma abordagem baseada em filtragem colaborativa integrada a mapas de saliência para a recomendação de imagens. Universidade Federal de Uberlândia, 2015. Citado 5 vezes nas páginas 19, 20, 23, 24 e 25.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado na página 28.

PROVOST, F.; FAWCETT, T.; BOSCATO, M. *Data Science Para Negócios*. ALTA BOOKS, 2016. ISBN 9788576089728. Disponível em: <https://books.google.com.br/books?id=c4lAvgAACAAJ>. Citado na página 29.

PU, P.; CHEN, L.; HU, R. Evaluating recommender systems from the user's perspective: survey of the state of the art. *User Modeling and User-Adapted Interaction*, Springer, v. 22, n. 4-5, p. 317–355, 2012. Citado na página 15.

RENDLE, S. et al. Neural collaborative filtering vs. matrix factorization revisited. In: *Fourteenth ACM Conference on Recommender Systems*. [S.l.: s.n.], 2020. p. 240–248. Citado na página 22.

SABITHA, S.; CHOUDHURY, T. et al. Proposed approach for book recommendation based on user k-nn. In: *Advances in computer and computational sciences*. [S.l.]: Springer, 2018. p. 543–558. Citado 2 vezes nas páginas 32 e 39.

SARWAR, B. et al. *Application of dimensionality reduction in recommender system-a case study*. [S.l.], 2000. Citado 2 vezes nas páginas 22 e 29.

SCHAFER, J. B.; KONSTAN, J. A.; RIEDL, J. E-commerce recommendation applications. *Data mining and knowledge discovery*, Springer, v. 5, n. 1, p. 115–153, 2001. Citado na página 19.

SOHAIL, S.; ANWAR, K.; SIDDIQUI, J. Machine learning based book recommender systems: A survey and new perspective. 11 2019. Citado na página 20.

SOHAIL, S. S.; SIDDIQUI, J.; ALI, R. Feature-based opinion mining approach (foma) for improved book recommendation. *Arabian Journal for Science and Engineering*, Springer, v. 43, n. 12, p. 8029–8048, 2018. Citado na página 32.

STRANG, G. Linear algebra and its applications, thomson learning. *Inc., London*, 1988. Citado na página 22.

VALDEZ, E. N. et al. A recommender system based on implicit feedback for selective dissemination of ebooks. *Information Sciences*, v. 467, p. 87–98, 10 2018. Citado na página 32.

WALEK, B.; FAJMON, P. A hybrid recommender system for an online store using a fuzzy expert system. *Expert Systems with Applications*, v. 212, p. 118565, 2023. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417422016293>. Citado na página 25.

WANG, D. et al. A content-based recommender system for computer science publications. *Knowledge-Based Systems*, Elsevier, v. 157, p. 1–9, 2018. Citado na página 32.

WANG, J.; VRIES, A. P. D.; REINDERS, M. J. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In: *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. [S.l.: s.n.], 2006. p. 501–508. Citado na página 20.

XIN, L. et al. Collaborative book recommendation based on readers' borrowing records. In: IEEE. *2013 International Conference on Advanced Cloud and Big Data*. [S.l.], 2013. p. 159–163. Citado na página 15.

ZHANG, S. et al. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 52, n. 1, p. 1–38, 2019. Citado na página 27.

ZURADA, J. M. *Introduction to artificial neural systems*. [S.l.]: West St. Paul, 1992. v. 8. Citado na página 27.