Teoria e Prática em Sistemas de Recomendação

Rogério Xavier de Azambuja[1,2,4], A. Jorge Morais[2,3], Vítor Filipe[4,5]

[1] Instituto Federal do Rio Grande do Sul (IFRS), 95174-274 Farroupilha/RS, Brasil
rogerio.xavier@farroupilha.ifrs.edu.br
[2] Universidade Aberta (UAb), 1269-001 Lisboa, Portugal
[3] LIAAD – INESC TEC, 4200-465 Porto, Portugal
Jorge.Morais@uab.pt
[4] Universidade de Trás-os-Montes e Alto Douro (UTAD), 5000-801 Vila Real, Portugal
[5] INESC TEC – INESC Tecnologia e Ciência, 4200-465 Porto, Portugal
vfilipe@utad.pt

Resumo

Nas últimas décadas a utilização da inteligência artificial tem sido frequente no desenvolvimento de aplicações computacionais. Mais recentemente a aprendizagem automática, especialmente pelo uso da aprendizagem profunda (*deep learning*), tem impulsionado o crescimento e ampliado o desenvolvimento de sistemas inteligentes para diferentes domínios. No cenário atual de crescimento tecnológico estão a surgir com maior frequência os sistemas de recomendação (*recommender systems*) com diferentes técnicas para a filtragem de informações em grandes bases de dados. Um desafio é prover a recomendação adaptativa para mitigar a sobrecarga de informações em ambientes *on-line*. Este artigo revisa trabalhos anteriores e aborda alguns dos aspectos teórico-conceptuais e teórico-práticos que constituem os sistemas de recomendação, caracterizando o emprego de redes neuronais profundas (*Deep Neural Network* – DNN) para prover a recomendação sequencial apoiada pela recomendação baseada em sessão.

Palavras-chave: sistemas de recomendação; filtragem de informação; recomendação com DNN; recomendação sequencial; recomendação baseada em sessão.

Title: Theory and Practice in Recommender Systems

Abstract: In recent decades, artificial intelligence use has been frequent in the computational applications development. More recently, machine learning, especially through the use of deep learning, has driven growth and expanded the intelligent systems development for different domains. In the current scenario of technological growth, the recommender systems appear with increasing frequency through their different techniques for information filtering in large datasets. It is a challenge to provide adaptive recommendation to mitigate information overload in online environments. This article reviews previous works and addresses some of the theoretical-conceptual and theoretical-practical aspects that constitute the recommender systems, characterizing the use of deep neural network (DNN) to provide sequential recommendation supported by session-based recommendation.

Keywords: recommender systems; information filtering; DNN recommendation; sequential recommendation; session-based recommendation.

1. Introdução

Na atualidade é encontrado um volume cada vez maior de informações *on-line*, dado o aumento no interesse por aplicações Web, bem como a oferta de produtos e serviços em escala mundial, como por exemplo em: *e-commerce*, *e-business*, *e-learning*, *e-tourism*, entre outros [Q. Zhang et al. 2021]. Inevitavelmente convive-se com a possibilidade de estar realizando diversas escolhas em tempo real e os sistemas de recomendação (*recommender systems* – RS) apresentam-se como uma estratégia eficaz para mitigar a sobrecarga de informações, oferecendo maior ponderação às decisões. É possível aos softwares apreenderem os comportamentos personalizados dos utilizadores e preverem as suas preferências de forma individualizada para atender às suas necessidades.

Os RS são classes de algoritmos com inteligência artificial que recomendam itens relevantes aos utilizadores, sendo utilizadas diferentes técnicas para a filtragem de informação que observam vários fatores, como por exemplo, o histórico das interações dos utilizadores. Investigações associando os RS com novos recursos tecnológicos e o desenvolvimento de projetos e de produtos em diferentes tecnologias estão em crescimento nos últimos anos [Shao et al. 2021]. Os RS têm evoluído desde o seu surgimento no início dos anos 1990 e mais recentemente têm sido associados com as técnicas de aprendizagem automática (*machine learning* – ML), tais como a aprendizagem por transferência, ativa ou por reforço, especialmente com a utilização da aprendizagem profunda (*deep learning* – DL) empregando redes neuronais profundas (*Deep Neural Network* – DNN) para a construção de recomendações relevantes através de modelos computacionais eficientes.

Grande parte dos RS têm sido implementados por meio de sistemas multi-agentes (*multi-agent system* – MAS) explorando a arquitetura *client-server* da Internet. Assim como em todos os sistemas, a arquitetura do software precisa ser bem estudada e definida para a sua efetiva utilização e para que possa representar ganhos aos desenvolvedores e aos utilizadores finais. O estudo e o aprofundamento em questões conceptuais e estruturais em RS sob diversas visões, são capazes de fornecer justificativa para cada uma das arquiteturas propostas e para qual cenário se enquadraria como mais benéfica. Além da computação, os fundamentos em RS incluem matemática, lógica, probabilidade estatística, filosofia, linguística e diversos conhecimentos associados a teoria da decisão.

Este artigo revisa trabalhos anteriores por meio de um estudo exploratório e apresenta nas próximas secções alguns aspectos teórico-conceptuais e teórico-práticos que constituem os RS. Foram revisadas publicações na ACM RecSys [n.d.] – considerada atualmente a maior conferência sobre o tema RS, também em categorias de publicações vinculadas a *Computer Science* e *Web Science* em Springer Link, IEEE Xplore, Google Scholar, Microsoft Academic e por bibliometria nos repositórios Clarivate Analytics Web of Science, EBSCO, Scopus Elsevier e outros. Procura-se caracterizar os RS apresentando os seus principais elementos e abordagens, especialmente o emprego recente de DNN em direção a geração da recomendação adaptativa em ambientes *on-line*. Pretende-se oferecer um contributo à investigação científica nesta área do conhecimento, sem a intenção de esgotar o assunto.

2. Aspectos Teórico-conceptuais

2.1. Breve Histórico

Anteriormente ao ano de 1990, são encontrados alguns relatos de experiências, estudos académicos como os dos professores G. Salton, M. J. McGill e C. Buckley na recuperação de informações textuais [Salton e McGill 1983; Salton e Buckley 1988], ou mesmo, experimentos envolvendo recomendações geradas algoritmicamente [Malone et al. 1987]. Contudo, em razão da maior relevância nos registros bibliográficos encontrados, a evolução dos sistemas de recomendação como um campo de investigação científica ocorreu a partir de 1990.

Pioneiramente, é preciso atribuir crédito (mesmo que de forma parcial) ao sueco Jussi Karlgren, que elaborou em outubro de 1990 uma álgebra para recomendação [Karlgren 1990]. Tratava-se de um relatório referente a um trabalho académico para melhorar a interação humano-computador utilizando modelos estatísticos em dois experimentos propostos. Em 1994 esta álgebra para recomendação foi publicada depois de revista e aprimorada [Karlgren 1994].

Um sistema de recomendação completo foi desenvolvido e publicado em 1992 por David Goldberg, David A. Nichols, Brian M. Oki e Douglas B. Terry com o intuito de auxiliar no aumento da capacidade e eficácia do processo de indicação de itens, reduzindo a obviedade em relações sociais entre seres humanos [Goldberg et al. 1992]. A motivação dos autores girava em torno de ser impulsionada a colaboração social para evitar o grande volume de documentos de notícias em circulação. A expressão "filtragem colaborativa" foi criada pelos autores numa abordagem para caracterizar um tipo específico de sistema, em que a seleção da informação era realizada automaticamente utilizando o auxílio humano pela colaboração entre os grupos interessados. Esta abordagem foi materializada no software Tapestry [Goldberg et al. 1992], conhecido até o presente momento como o primeiro RS computacional. O Tapestry foi projetado para recomendar por e-mail documentos extraídos de coleções de notícias para um grupo restrito de utilizadores na Xerox Palo Alto Research Center (Xerox PARC). Os criadores perceberam que os utilizadores recebiam muitos e-mails com documentos de notícias motivados pela facilidade que se tornou enviá-los eletrónicamente, porém sem nenhum ou pouco critério na filtragem do seu conteúdo.

Estudos subsequentes a 1992 encontrados em materiais de conferências e periódicos científicos, propuseram com êxito diferentes heurísticas originais e especializadas para a realização da filtragem de informação e obtenção de congruência em itens a serem recomendados aos utilizadores. Em 1994, destacou-se a criação do projeto GroupLens[1] pelo Instituto de Tecnologia de Massachusetts (MIT) conjuntamente com a Universidade de Minnesota apoiando investigações no tema RS, interagindo com empresas do setor produtivo e servindo como uma referência importante até os dias atuais. Entre as suas publicações pioneiras está "Uma Arquitetura Aberta para a Filtragem Colaborativa do *Netnews*" [Resnick et al. 1994], em que os autores mostraram preocupação com a sobrecarga de informações recebidas indiscriminadamente por e-mail e propuseram uma estrutura para que as pessoas pudessem avaliar as mensagens em seus *newsgroups*, levando em consideração neste caso, que haveria nova concordância por parte dos utilizadores com as classificações já realizadas anteriormente.

[1] Disponível em https://grouplens.org. Acedido em 12/08/2021.

Foram estudadas metodologias de avaliação das recomendações [Gunawardana e Shani 2009; Levandoski et al. 2011] e diferentes abordagens de filtragem de informação passaram a considerar as informações sobre o momento de recolha das preferências históricas dos utilizadores por meio de *feedbacks* que são recolhidos de forma explícita e implícita, bem como teorias de aproximação, mecanismos algorítmicos e estilos arquiteturais [Koren 2009; Castells et al. 2011; Ekstrand et al. 2011; Morais et al. 2012]. A matemática e a estatística são fortemente empregadas para gerar recomendações, pois a percepção humana pode ser diferente para itens semelhantes e vice-versa, por isso o cálculo da similaridade entre itens e utilizadores torna-se importante e, opostamente, há o cálculo da diversificação. Ambos tornam possível o tratamento eletrónico de grandes volumes de dados coletados que resultam em itens recomendados. Diversas medidas de similaridade, bem como a exploração da diversidade e outros métodos ainda mais complexos são encontrados na literatura procurando aliar desempenho computacional e confiabilidade à solução do problema de recomendação [Liu et al. 2014].

Mais recentemente, são utilizados métodos de aprendizagem automática, com crescimento nos últimos anos da aprendizagem profunda [Shao et al. 2021; Q. Zhang et al. 2021; Hiriyannaiah et al. 2020; Zhang et al. 2019], além de outros aplicados em diferentes domínios para a filtragem de grandes volumes de dados. Mecanismos algoritmos de autoatenção tem sido empregados no problema de recomendação com DNN, em que os dados de entrada são considerados pelos algoritmos por diferentes formas e não influenciam igualmente cada saída do modelo computacional [Li et al. 2017]. Conforme ilustrado na Figura 1, o aumento da interatividade entre utilizadores e itens em ambientes *on-line* tem sido utilizada para gerar dados empregados na aprendizagem automática, previsão e personalização da experiência do utilizador em vários domínios de aplicações [Gharahighehi e Vens 2021]. Ainda, a recomendação sequencial apoiada pela recomendação baseada em sessão estão ganhando foco em investigações que tratam a relação entre os interesses dos utilizadores nas dinâmicas sequenciais de curto prazo e de longo prazo [Jannach et al. 2020].

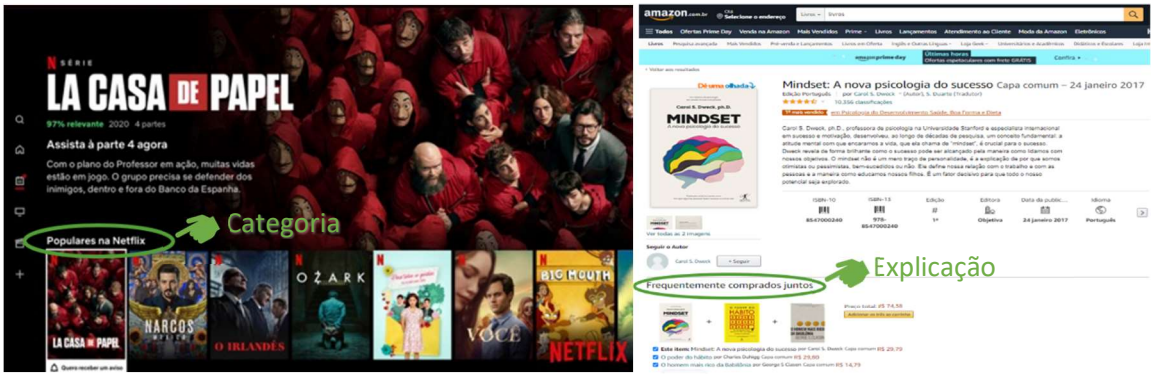


Figura 1. Exemplo de categoria e explicação da recomendação, retirado dos portais oficiais Netflix.com e Amazon.com em 12/08/2021.

Investigações e publicações de relevância no tema RS difundiram-se pelo mundo com grande impacto na Europa, América e em países asiáticos como China e Índia, bem como em outros [Shao et al. 2021]. Também é destacado o surgimento de plataformas centralizadoras de conteúdos no tema como Recommender-systems.com[2], Beta-recsys[3], entre outras. Empresas como Netflix, Amazon, Google (Ex. YouTube, Gmail, motor de busca, ...), entre outras já

[2] Disponível em https://recommender-systems.com. Acedido em 12/08/2021.
[3] Disponível em https://beta-recsys.readthedocs.io. Acedido em 12/08/2021.

apresentam recomendações na oferta de muitos dos seus produtos utilizando diferentes abordagens que serão detalhadas a seguir.

2.2. Definição

O objeto de investigação em RS são as técnicas de filtragem de informação que tratam basicamente de como fazer recomendações relevantes de itens para utilizadores [Falk 2019]. O modelo mais simples pode ser expresso por uma matriz contendo as relações para a fatorização: M = $\boldsymbol{U}$ x $\boldsymbol{I}$, em que $\boldsymbol{U}$ denota o conjunto de utilizadores $\{u_1, ..., u_{|U|}\}$ e $\boldsymbol{I}$ o conjunto de itens $\{i_1, ..., i_{|I|}\}$. É possível pensar que aleatoriamente poderia ser sorteado os valores para cada par M(u, i) e rapidamente se completaria a matriz M com as classificações para a geração das recomendações desejadas. Isso seria computacionalmente eficaz, entretanto não garante a premissa de fazer boas recomendações. As avaliações de preferências (*ratings*) disponíveis no momento contextual ($\boldsymbol{U} \times \boldsymbol{I}$), são obtidas por *feedbacks* explícitos e implícitos. As preferências podem ser esparsas (*cold start problem*), sofrer alterações temporais ou serem estimadas por mecanismos de previsão. Sumariamente, em RS estudam-se técnicas para se obter as preferências do utilizador (completar a matriz M ou outra estrutura de dados utilizada) de forma satisfatória para a geração da recomendação com o menor esforço computacional não proibitivo. A estrutura básica arquitetural é ilustrada na Figura 2, em que o utilizador final pode fazer parte do conjunto de utilizadores iniciais, exceto na primeira vez, quando for um novo utilizador.

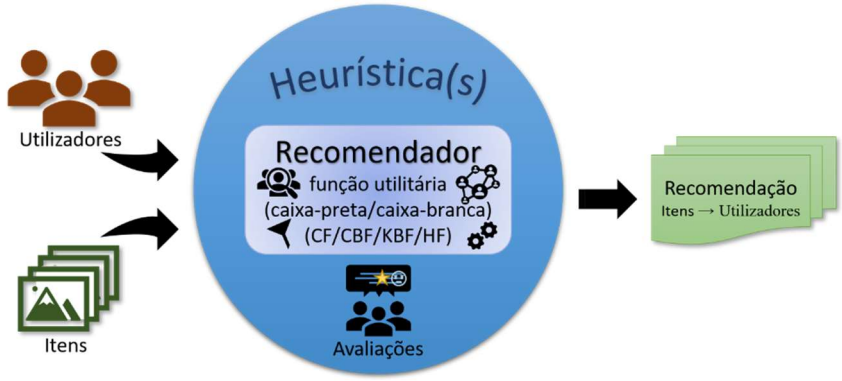


Figura 2. Estrutura básica empregada em sistemas de recomendação.

Várias situações precisam ser levadas em consideração para se obter êxito nas recomendações. Algumas delas estão relacionadas com os dados e medidas utilizadas, além dos algoritmos empregados na criação de modelos computacionais e nas metodologias de avaliação dos RS, sendo ambos abordados com maior detalhe na Secção 3. Dada a fatorização dos conjuntos $\boldsymbol{U}$ e $\boldsymbol{I}$ expressa pela equação (1), busca-se uma função utilitária que leve a um conjunto *Top*-N de recomendações relevantes $\boldsymbol{R}$ = $\{rec_1, ..., rec_N\}$, em que N $\in \mathbb{N} > 0$ e $\boldsymbol{R} \subset \boldsymbol{I}$:

$$f: \boldsymbol{U} \times \boldsymbol{I} \rightarrow \boldsymbol{R} \quad (1)$$

Os conjuntos de utilizadores $\boldsymbol{U}$ e de itens $\boldsymbol{I}$ comumente são representados por metadados ricos, formados por dados estruturados contínuos como valores, por exemplo, também por dados estruturados categóricos como o género de um filme, sexo, etc. ou ainda por dados não estruturados, tais como: textuais, áudios, vídeos, etc. Além da veracidade dos dados, alguns fatores importantes são a escalabilidade do volume nos conjuntos de dados, a variabilidade dos

metadados e a sua taxa de atualização. A Figura 3 ilustra um exemplo com uma possível combinação paramétrica num RS.

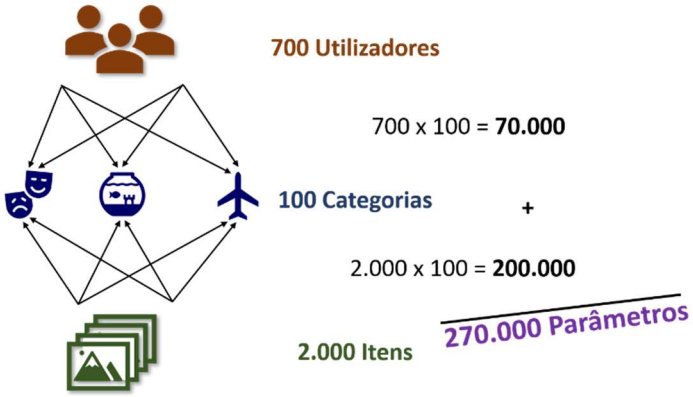


Figura 3. Simulação paramétrica de um problema de recomendação.

Dependendo do domínio da aplicação, diversas características podem ser representadas e normalmente são utilizadas instâncias de matrizes multidimensionais e *embeddings* (abordados na subseção 3.1) no processamento da recomendação de itens para cada utilizador, visando maximizar a função utilitária definida [Q. Zhang et al. 2021 como citado em Adomavicius e Tuzhilin 2005], assim expressa:

$$\forall u \in \boldsymbol{U}, \arg\max_{i \in \boldsymbol{I}} f(u, i) \tag{2}$$

Uma situação encontrada e de grande relevância contextual em RS é o problema chamado de "arranque a frio" (*cold start*) quando há escassez de dados para a recomendação gerando matrizes esparsas e então várias estimativas precisam de ser realizadas. Isso acontece invariavelmente em momentos iniciais de execução dos sistemas ou mesmo, quando do ingresso de novos utilizadores e/ou de novos itens no RS. Alguns autores estão propondo trabalhos na linha de recomendações coletivas para grupos de utilizadores a fim de mitigar o *cold start problem* [Dara et al. 2020; Felfernig et al. 2018].

São encontradas na literatura de referência diversas definições que se complementam em torno dos RS, a maioria delas concordam com as apresentadas a seguir, tendo algumas publicações como ponto de arranque [Falk 2019; Ludewig e Jannach 2018; Taghavi et al. 2018; Aggarwal 2016; Ricci et al. 2015; Adomavicius e Tuzhilin 2005]. Desta forma, um sistema de recomendação calcula e oferece conteúdo relevante para o utilizador com base no conhecimento deste, no próprio conteúdo e nas interações entre o utilizador e o item. Formalmente, o problema de recomendação consiste na definição de uma função utilitária para recomendar ao utilizador um ou mais itens com as mais altas avaliações estimadas e ranqueadas por um *score* de saída obtido para cada item [Adomavicius e Tuzhilin 2005]. Outras definições importantes e complementares também são encontradas e sumarizadas no Quadro 1.

Quadro 1: Algumas definições importantes em Sistemas de Recomendação.

Utilizador *(user)* $\boldsymbol{U} = \{u_1, ..., u_{\lvert U \rvert}\}$	Alguém que faça uso de um sistema que lhe proporcionará interação com itens. Semelhante a um ator num caso de uso na linguagem de modelação unificada (UML). Uma entidade que forma o conjunto de dados iniciais ***U***.
Item (*item*) $\boldsymbol{I} = \{i_1, ..., i_{\lvert I \rvert}\}$	Algo de interesse do utilizador. Uma entidade que forma o conjunto de dados iniciais ***I***.
Recomendação (*recommendation*) $\boldsymbol{R} = \{rec_1, ..., rec_N\}$	Os N itens mais relevantes (*Top*-N), em que $N \in \mathbb{N} > 0$ e $\boldsymbol{R} \subset \boldsymbol{I}$.
Relevância (*relevancy*) $r_{u,i} = f(u, i)$	Um *score* para cada item de acordo com o que é mais proeminente para o utilizador em um dado momento. Relevância é uma função de contexto, demográfica e/ou classificatória.
Previsão (*prediction*) $\hat{r}_{u,i} = f(u, i)$	Uma estimativa de relevância do item para o utilizador.
Personalização (*personalization*)	A forma de apresentação da recomendação.
Perfil de gosto (*taste profile*) *taste* = {a, b, c}	Uma ordenação dos itens formando uma lista de termos caracterizadores e seus valores de relevância.
Colaboração (*collaboration*)	Estratégia realizada conjuntamente para produzir algo.
Filtragem (*filtering*)	Procedimento para selecionar utilizadores ou itens de um conjunto deles.
Similaridade (*similarity*) *Similarity(x,y)* = #	A medida de quão um elemento se aproxima de outro. Pode ser considerada ou não a semelhança física entre os elementos (*x*, *y*, ...). Normalmente são obtidos valores entre 0 e 1, representando de baixa até alta similaridade, respectivamente. São exemplos as medidas calculadas por Cosseno, Pearson, Jaccard, Tanimoto, Euclidiana, Manhattan, Hammann, BiLRP, entre outras.
Diversidade (*diversity*) *Diversity(x,y)* = #	A medida de quão um elemento é distante de outro. O contrário da similaridade, representa a diferença entre os elementos (*x*, *y*, ...). Normalmente faz parte da equação da diversidade a expressão (1 - *Similarity(x, y)*).
Diversificação (*diversification*)	Procedimento que utiliza a diversidade para a criação de um conjunto variado de elementos.
Novidade (*novelty*) *Novelty(**R**)* = #	A medida de quão bem são recomendados itens desconhecidos (que não seriam encontrados por *default*) para os utilizadores ou em relação a lista de itens recomendados no passado.
***S*erendipidade (*serendipity*)** *Serendipity(**R**)* = #	A medida de quão bem são recomendados itens que são inesperados ou mesmo, de grande satisfação para os utilizadores.
Fatorização de matriz (*matrix factorization*)	O processo de transformar numa matriz o produto de duas ou mais matrizes.

Entidade (*entity*)	Um conjunto de objetos (normalmente reais) que são representados por meio da modelação dos seus atributos. Também conhecido como um nó em uma rede.
Rede relacional (*relational network*) RN(V, A) V={ *e* | *e* é uma entidade} A={ (v,w) | <v liga w> }	Um conjunto de dados representados em um grafo no qual os vértices ou nós correspondem às entidades e as arestas ou ligações entre os nós correspondem às relações entre as entidades.
Vizinhança (*neighborhood*) N'(V', A')	Os nós em uma rede que estão vinculados em um conjunto de dados relacional. Um subgrafo na rede relacional, em que N'(V', A') ⊂ RN(V, A). Também formam uma vizinhança as amostras que são semelhantes entre si com base em certa métrica, mesmo em um conjunto de dados não relacional.
Sessão (*session-based*)	Uma sessão monolítica registra uma execução de um sistema e normalmente possui um intervalo de tempo. Por exemplo na Web, uma sessão é iniciada quando um utilizador acessa um determinado *website*, continua durante a navegação e termina quando se encerra o acesso.
Intervalo de Tempo (*time interval*) t	Duração de um segmento de tempo sem referenciar quando este começa ou termina (segundos, minutos, horas, dias, semanas, meses, anos, ...). Na literatura são encontradas as dinâmicas sequenciais de curto prazo – normalmente formada por uma única sessão com duração de alguns segundos, minutos até algumas horas e de longo prazo – normalmente formada por mais de uma sessão durante alguns dias, semanas, meses, ...

2.3. Taxonomia

O recomendador ilustrado na Figura 2 pode adotar modelos computacionais distintos para computar as recomendações. Uma primeira distinção de acordo com Herlocker et al. [2000] formam dois tipos: modelos caixa-preta, que procuram não revelar o processo pelo qual produzem a recomendação, explicitando apenas quais são as entradas e quais serão as saídas do recomendador e modelos caixa-branca, que tornam visíveis as técnicas utilizadas de forma a justificar a recomendação gerada.

Além disso, os RS podem ser divididos em diferentes abordagens de acordo com alguns critérios definidos na filtragem de informação empregada. Principalmente são encontradas as seguintes abordagens: colaborativa, baseada em conteúdo, baseada em conhecimento ou mesmo híbrida. Contudo, diversas taxonomias são encontradas na literatura marcando evoluções nos RS ao longo do tempo. Maheswari et al. [2019] e outros autores apresentam uma taxonomia bastante difundida que é ilustrada na Figura 4. Entretanto, mesmo sendo representativa das principais abordagens encontradas e muito usual, não há consenso por apenas esta classificação. Também são encontrados, por exemplo, RS baseados em grupos/comunidade, baseados em demografia, sensível ao contexto, entre outras formas de representação, muitas delas especializadas.

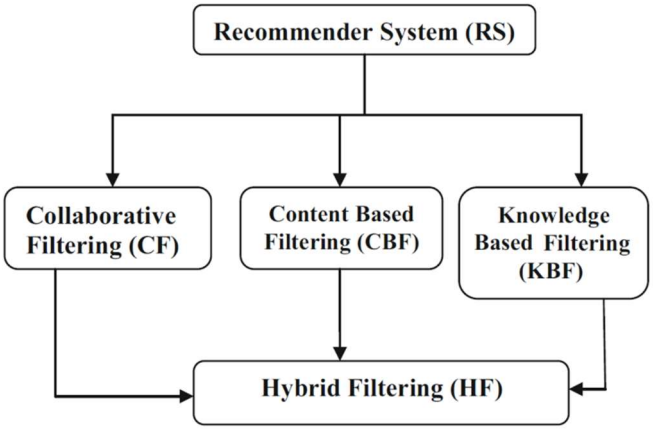
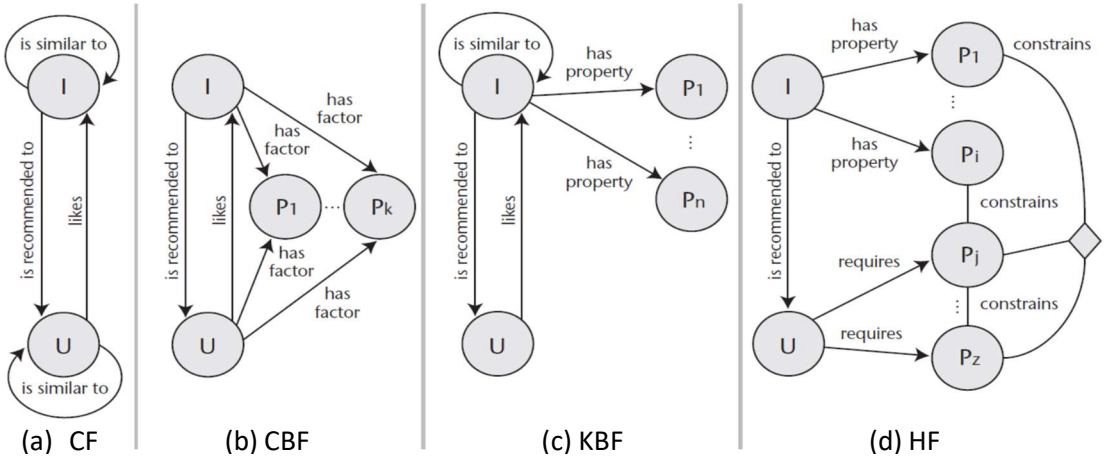


Figura 4. Taxonomia dos sistemas de recomendação, retirada de Maheswari et al. [2019].

São encontradas diversas definições complementares para as quatro principais abordagens dos sistemas de recomendação [Q. Zhang et al. 2021; Maheswari et al. 2019; Zhang et al. 2019], a maioria delas concordam com as apresentadas a seguir. Para descrevê-las sumariamente, serão utilizados os arquétipos ilustrados na Figura 5, que abrem as caixas referidas na Figura 4 em fluxos para uma visualização mais detalhada da utilização dos conceitos.



Em que, U = conjunto de utilizadores, I = conjunto de Itens e P = propriedade abstrata (ex. categoria).

Figura 5. Arquétipos dos sistemas de recomendação, retirados de Friedrich e Zanker [2011].

As principais abordagens de filtragem de informações em RS podem ser assim caracterizadas:

- A **filtragem colaborativa (CF)** procura construir recomendações a partir das interações históricas do utilizador-item, sejam por meio de dados obtidos em *feedbacks* explícitos ou implícitos. São consideradas as preferências dos utilizadores para encontrar outros utilizadores semelhantes (vizinhos de categoria) e baseadas nesta associação de utilizadores, realizar a recomendação assumindo que possam gostar dos mesmos itens. As técnicas colaborativas utilizadas podem ser subclassificadas em:
    - **CF baseada em memória**: com a utilização do conceito de vizinhança por meio de métodos baseados no utilizador e baseados em itens; e
    - **CF baseada em modelo**: com a aprendizagem e o tratamento dos gostos e relações de preferências dos utilizadores por itens.

- A **filtragem baseada em conteúdo (CBF)** concentra-se principalmente nas comparações entre itens semelhantes (similaridade) ou dissimilares (diversidade) em que os utilizadores interagiram anteriormente. As recomendações de itens semelhantes aos itens que os utilizadores já gostaram anteriormente é altamente dependente do conjunto de metadados dos próprios itens (*features*) e dos gerados por interações históricas dos utilizadores com itens semelhantes.
- A **filtragem baseada em conhecimento (KBF)** procura identificar conhecimentos específicos do domínio em fontes internas (tendências encontradas nos dados) ou mesmo externas para realizar a recomendação. São consideradas as sugestões e os gostos proporcionados pelos utilizadores com base nas suas interações históricas, ou seja, o modelo deve levar em consideração a base de conhecimento do domínio para gerar novas recomendações de itens para os utilizadores. Diversas informações auxiliares além das avaliações (*ratings*) em diferentes formatos (textos, imagens, vídeos, ...) podem ser exploradas para extrair conhecimento do domínio.
- A **filtragem híbrida (HF)** refere-se ao modelo que combina dois ou mais tipos de estratégias para melhorar a qualidade da recomendação. Primeiramente, a filtragem é realizada pelos métodos individualizados (normalmente CF e CBF ou ainda KBF), logo após o resultado é combinado em vários *clusters*, sendo utilizadas funções lineares para gerar coeficientes de similaridade e restrições. Por fim, os *clusters* similares identificados são unidos e utilizados para encontrar os utilizadores e/ou itens semelhantes e prover as recomendações de itens para os utilizadores.

3. Aspectos Teórico-práticos

3.1. Manipulando Dados e Metadados

Os dados em diferentes tipos e formatos podem ser obtidos por processos distintos de recolha em bases de dados e em interações *on-line* dos utilizadores em que podem ser utilizadas sessões monolíticas, que registam uma execução de um sistema, sendo muito utilizada no ambiente Web. Basicamente, além dos conjuntos iniciais $\boldsymbol{U}$ e $\boldsymbol{I}$, as avaliações de preferências são geradas por *feedbacks* proporcionados pelos utilizadores de forma explícita e implícita. O *feedback* explícito constitui dados ditos diretos e quantitativos, ou seja, coletados quando há a interação intencional do utilizador com o item, como por exemplo numa avaliação realizada atribuindo pesos escalares de 0 a 5 ou atribuindo a proporção de satisfação binária (bem/mal) ou ainda representadas por estrelas, etc. O *feedback* implícito constitui dados que são obtidos indiretamente das interações do utilizador ao serem coletados através do registo de acessos realizados, itens pesquisados e selecionados, tempo desprendido em cada ação, etc., como por exemplo em uma navegação Web em que os cliques realizados podem ser mapeados para a identificação de preferências do utilizador.

Os feedbacks explícitos são encontrados em menor número, já os *feedbacks* implícitos são abundantes, pois identifica-se na prática, que a navegação em conteúdo é proporcionalmente muito maior em quantidade em relação as avaliações e classificações realizadas intencionalmente pelos utilizadores. Existem diferentes bases de dados disponibilizadas em

repositórios, tais como as usuais MovieLens[1], Amazon[4], Diginetica[5], Yoochoose[5], Github[6], Printerest[7], Gowalla[8], Taobao[9], entre outras. Estas são amplamente conhecidas e utilizadas para a geração de indicadores por métricas para as avaliações e comparações de modelos computacionais de recomendação. Além disso, os *feedbacks* implícitos permitem aos RS a produção e adequação de recomendação em tempo real pelo ajuste paramétrico e empregando técnicas de filtragem para mitigar a sobrecarga de informações em ambientes *on-line*.

Os metadados são estruturas que descrevem os dados utilizados no âmbito de uma aplicação específica, para que possam ser representados e manipulados (persistência e recuperação, por exemplo). Para representar os dados utilizados em RS e permitir a exploração computacional das propriedades abstratas (P_1, ..., P_n, ..., P_z) ilustradas na Figura 5 e as restrições de domínios por diferentes técnicas matemáticas e estatísticas, por exemplo, é comumente utilizada a representação por *embeddings*. Trata-se da representação de um conjunto de dados num espaço vetorial multidimensional em que é realizado o mapeamento de objetos discretos em sequências de números contínuos. Normalmente, a representação por *embeddings* é realizada num espaço vetorial de baixa dimensão procurando capturar a relação vetorial encontrada nos dados utilizados num espaço dimensional maior. A Figura 6 ilustra uma possível representação vetorial bidimensional.

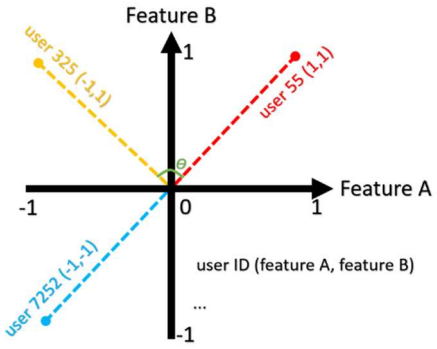


Figura 6. Exemplo de um *embedding* vetorial bidimensional.

No exemplo ilustrado na Figura 6, são encontradas as relações entre três utilizadores (identificados pelos IDs 55, 325 e 7252) e duas características verificadas (A e B), produzindo os vetores (1,1), (-1,1) e (-1,-1). Com a construção de *embeddings* é possível calcular, por exemplo, a similaridade entre um utilizador e outro, entre itens distintos ou mesmo, calcular a diversificação de itens num determinado domínio de aplicação, isso por diferentes técnicas. *Embeddings* também podem ser encontrados automaticamente através de aprendizagem automática, em que o sistema apreende os gostos e as preferências dos utilizadores, por exemplo.

3.2 Calculando Similaridade e Diversidade

[4] Disponível em http://jmcauley.ucsd.edu/data/amazon/. Acedido em 15/08/2021.
[5] Disponível em https://beta-recsys.readthedocs.io/en/latest/notes/datasets.html. Acedido em 15/08/2021.
[6] Disponível em https://github.com/John-K92/Recommendation-Systems-for-GitHub. Acedido em 15/08/2021.
[7] Disponível em https://sites.google.com/site/xueatalphabeta/academic-projects. Acedido em 15/08/2021.
[8] Disponível em http://snap.stanford.edu/data/loc-gowalla.html. Acedido em 15/08/2021.
[9] Disponível em https://tianchi.aliyun.com/datalab/dataSet.html?dataId=649. Acedido em 15/08/2021.

As medidas de similaridade têm se mostrado muito eficazes para tratar grandes volumes de dados, sobretudo quando as matrizes de avaliação dos itens por utilizadores são esparsas. Bem como a diversidade, tipicamente a medida oposta da similaridade, que também pode ser utilizada para indicar o quão diferentes entre si são os itens recomendados.

A Tabela 1 mostra alguns exemplos de medidas para o cálculo da similaridade, tais como: Cosseno, Pearson, Jaccard e Tanimoto, entretanto há outras medidas como: Euclidiana, Manhattan, Hammann, BiLRP e o *Deep Visual Ensemble Similarity Metric* (DVESM) proposto em Hiriyannaiah et al. [2020], em que as relações representadas por *embeddings* podem ser encontradas por ML em procedimentos de DL e aprimorar as preferências dos utilizadores a partir dos conjuntos de dados iniciais formados por todos os utilizadores e itens.

Tabela 1. Algumas medidas de similaridade, retirada de Maheswari et al. [2019].

Similarity measure	*Expressions*	
Cosine similarity	$\cos(\theta) = \frac{A+B}{\lvert\lvert A\rvert\rvert\lvert\lvert B\rvert\rvert} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2}\times\sqrt{\sum_{i=1}^{n}(B_i)^2}}$	(3)
Pearsson´s coeficiente	$P_{x,y} = \frac{\sum_{i=1}^{n}((x_i-\overline{x})(y_i-\overline{y}))}{\sqrt{\sum_{i=1}^{n}((x_i-\overline{x}))}*\sqrt{\sum_{i=1}^{n}((x_i-\overline{x})(y_i-\overline{y}))}}$	(4)
Jaccard similarity	$J(x,y) = \frac{\sum_i \min(x_i, y_i)}{\sum_i \max(x_i, y_i)}$	(5)
Tanimoto similarity	$f(A,B) = \frac{A.B}{\lvert A\rvert^2 + \lvert B\rvert^2 - A.B}$	(6)

Sendo que *A* e *B* ou *x* e *y* nas equações da Tabela 1 são as representações vetoriais dos metadados (utilizadores, itens, avaliações, ...) a ser calculada a similaridade entre eles. Por exemplo, a métrica de similaridade Cosseno (3) é uma medida do ângulo entre *A* e *B* num plano cartesiano (Ex. ângulo *Θ* na Figura 6) e é encontrado um valor resultante entre 0 e 1, em que se o ângulo entre *A* e *B* for 0º, a similaridade será igual a 1 (alta similaridade) e se o ângulo calculado for maior que 0º, a similaridade diminui proporcionalmente de 1 até 0. Diversas outras medidas de similaridade são encontradas na literatura e alguns métodos utilizados permitem alternar o cálculo entre mais de uma métrica de similaridade, sobretudo com o emprego de DNN [Eberle et al. 2020; Hiriyannaiah et al. 2020].

Quando a obtenção de itens por similaridade não apresentar bons resultados, por exemplo quando o ranqueamento *Top*-N é constituído por itens muito similares e, portanto com pouquíssimas diferenças entre si, muitas vezes torna-se importante alterar a variabilidade da amostra através do cálculo da diversidade em busca de melhores recomendações. Smyth e McClave [2001, p. 349] definem a diversidade de um conjunto de casos c_1, ..., c_n, como a diferença média entre todos os pares de casos no conjunto de casos, expressa pela fórmula:

$$Diversity_{(c_1,..,c_n)} = \frac{\sum_{i=1..n}\sum_{j=i..n}\left(1 - Similarity_{(c_i, c_j)}\right)}{\frac{n}{2} * (n-1)} \quad (7)$$

Algumas heurísticas ressaltam a importância da utilização de estratégias diferentes na obtenção de melhores resultados na solução do problema de recomendação. De acordo com Smyth e McClave [2001], é possível trocar em alguns casos as perdas de similaridade (como as

verificadas em CBF, por exemplo), por ganhos de diversidade. Além do cálculo da similaridade e da diversidade, podem ser utilizadas ainda outras medidas, como o cálculo da novidade (*novelty*) para identificar itens que não seriam facilmente encontrados por utilizadores, sobretudo disponibilizados por diversificação, popularidade (*popularity*) ou ainda, a medida de serendipidade (*serendipity*) que irá requerer mais dados auxiliares para a recomendação de itens considerados inesperados ou de grande satisfação aos utilizadores [Herlocker et al. 2004].

3.3. Algoritmos com Redes Neuronais Profundas

Os RS são amplamente difundidos em diferentes domínios de aplicações com a utilização de algoritmos que realizam filtragens de informação em grandes volumes de dados. Normalmente um algoritmo padrão de filtragem de informação implementa as equações (1) e (2) por diferentes técnicas e abordagens, executando ao menos, os passos relacionados abaixo:

> Passo 1: Obter os conjuntos de metadados de utilizadores $\boldsymbol{U}$ e de itens $\boldsymbol{I}$
>
> Passo 2: Obter/prever os pesos de avaliações e/ou relações $\boldsymbol{U} \times \boldsymbol{I}$ (ex. classificação)
>
> Passo 3: Processar a função utilitária definida f: $\boldsymbol{U} \times \boldsymbol{I} \rightarrow \boldsymbol{R}$ (ex. encontrar os vizinhos)
>
> Passo 4: Recomendar *Top*-N itens para cada utilizador

De acordo com Shao et al. [2021], que analisaram as publicações relacionadas ao tema '*recommender systems*' no repositório Clarivate Web of Science no período de 2009 a 2018, é constatada a utilização de sistemas multi-agentes na modelação e implementação de RS e o emprego de *deep learning* está em crescimento nos últimos anos, especialmente a partir de 2016. O DL pode ser utilizado em RS, por exemplo, para a extração automatizada de características dos metadados que representam os conjuntos de utilizadores e de itens. Por interações lineares e não lineares entre os conjuntos de utilizadores e de itens, em que uma estrutura mais complexa das preferências dos utilizadores pode ser elaborada e computacionalmente ser extraída dos dados de alta dimensão (ex. *embeddings*). Além disso, vários algoritmos de recomendação tradicionais e bem conhecidos na literatura podem ser transformados em uma ou mais camadas numa DNN. De acordo com Zhang et al. [2019, p. 2]:

> "A aprendizagem profunda é capaz de capturar com eficácia as relações não lineares e não triviais do utilizador-item, e permitir a codificação de abstrações mais complexas como representações de dados nas camadas superiores. Além disso, ele captura os relacionamentos intrincados dentro dos próprios dados, de fontes de dados acessíveis abundantes, como informações contextuais, textuais e visuais."

Especificamente, DL formando DNNs é um tipo de aprendizagem automática, uma técnica que permite o aprimoramento dos sistemas computacionais por meio da experiência e dos dados. As redes neuronais ditas tradicionais com uma ou duas camadas ocultas requerem mais informações sobre os recursos de entrada para realizar a seleção destes. Por outro lado, as DNNs não precisam de nenhuma ou de pouca informação sobre os recursos de entrada, sendo capazes de processarem a seleção destes nas primeiras camadas e ao longo de suas camadas ocultas, processarem o ajuste e a seleção do modelo ideal. A lógica de funcionamento empregada em algoritmos com DNN é a seguinte: com base no que já se sabe é possível prever ou inferir algo e se algo sair errado, é possível quantificar o erro em perda e melhorar o modelo – que representa as hipóteses de como o mundo funciona [Skansi 2018], até satisfazer os critérios de acurácia e frequência.

DNNs podem ser interpretadas como camadas empilhadas de transformações não lineares, apreendendo representações hierárquicas de *features*. Em cada camada da rede neuronal, as funções de ativação com suas características específicas detalhadas em A. Zhang et al. [2021, p. 132–137], podem ser utilizadas para cada contexto. São exemplos de função de ativação: ReLU, pReLU, Sigmoid, Tanh, mais recentemente o *Gradient Class Activation Map* (Grad-CAM) utilizado em Greenspan et al. [2020], entre outras. Existem também as funções de otimização, que procuram em cada camada diminuir o erro entre os resultados obtidos (treino) quando comparados posteriormente aos resultados desejados (teste) para aumentar a acurácia e melhorar o desempenho algorítmico, tais como: Adam, Adagrad, Adabound, RMSprop, SGD com Momentum, dentre outras.

Zhang et al. [2019] propõem uma classificação considerando o tipo de técnica utilizada para reunir em grupos e organizar as arquiteturas trabalhadas em publicações eleitas pelos autores como as mais relevantes até 2019 com o emprego de DL em RS. O esquema proposto é ilustrado na Figura 7.

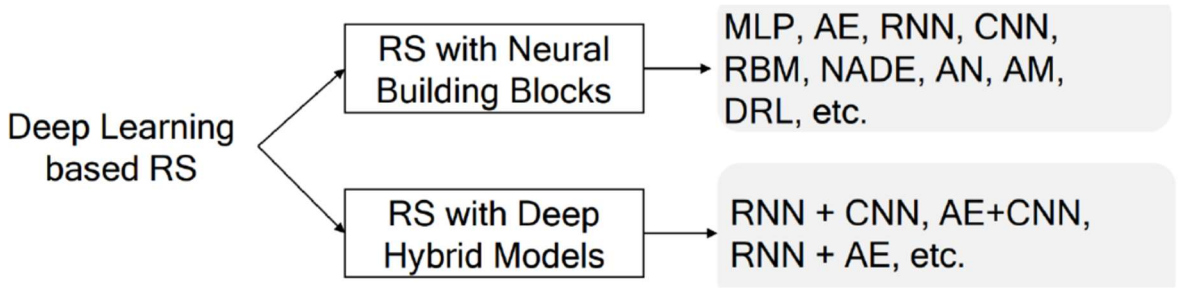


Figura 7. Categorias de redes neuronais profundas baseadas em sistemas de recomendação, retirada de Zhang et al. [2019].

Sumariamente, os RS com blocos de construção neuronais consideram que a técnica de DL empregada determina o modelo de recomendação a ser adotado. Por exemplo, ao ser utilizada a arquitetura multicamadas (*Multilayer Perceptron* – MLP) pode-se modelar as interações não lineares entre utilizadores e itens, já com redes neuronais convolucionais (*Convolutional Neural Network* – CNN) torna-se possível a extração de *embeddings* de dados heterogéneos ou ainda redes neuronais recorrentes (*Recurrent Neural Network* – RNN) para gerar recomendações *on-line* com dados sequenciais como séries temporais, textos e áudios. Os RS com modelos híbridos profundos acabam por utilizar mais de uma técnica de DL, tornando possível a combinação de mais de um bloco de construção. Isso permite a união de técnicas em um modelo mais robusto com a complementação entre si dos modelos utilizados, pois podem existir várias combinações que se tornam possíveis. Por exemplo: CNN + *Autoencoders* (AE) ou CNN explorando a aprendizagem por reforço (*Deep Reinforcement Learning* – DLR), entre outras combinações.

Como um exemplo concreto, a classe apresentada no Código-fonte 1 (*Neural Collaborative Filtering* – NCF), define um modelo de filtragem colaborativa neuronal utilizando a arquitetura MLP em cinco camadas. Nesta implementação na linguagem de programação Python foi utilizado o *framework* proposto em He et al. [2017] que emprega ainda o coeficiente de Jaccard como medida de similaridade entre dois utilizadores na matriz de fatorização.

Código-fonte 1. Treinamento neuronal utilizado filtragem colaborativa baseada em modelo, adaptado de Loy [2020].

```
# carregamento da classe com o modelo de treinamento
class NCF(pl.LightningModule):
    """ Neural Collaborative Filtering (NCF)
        Args:
            num_users (int): Number of unique users
            num_items (int): Number of unique items
            ratings (pd.DataFrame): Dataframe containing the ratings for training
            all_itemsIds (list): List containing all itemsIds (train + test)
    """
    def __init__(self, num_users, num_items, ratings, all_itemsIds):
        super().__init__()
        self.user_embedding = nn.Embedding(num_embeddings=num_users, embedding_dim=8)
        self.item_embedding = nn.Embedding(num_embeddings=num_items, embedding_dim=8)
        self.fc1 = nn.Linear(in_features=16, out_features=64)
        self.fc2 = nn.Linear(in_features=64, out_features=32)
        self.output = nn.Linear(in_features=32, out_features=1)
        self.ratings = ratings
        self.all_itemsIds = all_itemsIds

    def forward(self, user_input, item_input):

        # Pass through embedding layers
        user_embedded = self.user_embedding(user_input)
        item_embedded = self.item_embedding(item_input)

        # Concat the two embedding layers
        vector = torch.cat([user_embedded, item_embedded], dim=-1)

        # Pass through dense layer
        vector = nn.ReLU()(self.fc1(vector))
        vector = nn.ReLU()(self.fc2(vector))

        # Output layer
        pred = nn.Sigmoid()(self.output(vector))

        return pred

    def training_step(self, batch, batch_idx):
        user_input, item_input, labels = batch
        predicted_labels = self(user_input, item_input)
        loss = nn.BCELoss()(predicted_labels, labels.view(-1, 1).float())
        return loss

    def configure_optimizers(self):
        return torch.optim.Adam(self.parameters())

    def train_dataloader(self):
        return DataLoader(TrainDataset(self.ratings, self.all_itemsIds), batch_size=512, num_workers=0)
```

Neste trecho de codificação apresentado no Código-fonte 1, são utilizados como tensores as avaliações de filmes (parâmetro *ratings*) e os Ids de todos os filmes (parâmetro *all_itemsIds*) com o objetivo de gerar recomendações com base nas relações históricas dos utilizadores com os itens. O método construtor da classe NCF(*num_users, num_items, ratings, all_itemsIds*), destacado em quadro na cor vermelha (linhas 2-10), utiliza as duas primeiras camadas para a detecção de *embeddings* de utilizadores e de itens num espaço vetorial com oito dimensões, duas camadas intermediárias densas utilizando a função de ativação ReLU e, por fim, uma camada de saída com ativação por uma função Sigmoid, bem como o otimizador de resultados Adam, que apresenta bom desempenho em segmentações e classificações. Um exemplo completo pode ser encontrado em Loy [2020].

O resultado final do exemplo concreto que instancia a classe NCF será um indicador percentual de assertividade do modelo computacional para as *Top*-N recomendações de itens previstos para cada utilizador (métricas de avaliação detalhadas na subseção 3.4), neste caso referente ao

percentual de sucesso na recomendação de filmes, sendo utilizada neste exemplo concreto a base de dados MovieLens[1]. Podem ser utilizadas diferentes bases de dados para a realização de experimentos com o Código-fonte 1, dividindo as avaliações e/ou relações $\boldsymbol{U} \times \boldsymbol{I}$ em dois grupos: treino do modelo e teste de assertividade. Normalmente na utilização de DL é recomendada uma divisão aproximada em torno de 70% e 30% ou 80% e 20% respectivamente, a ser reajustada condicionalmente.

Em RS não fará sentido o treino ser realizado numa população de utilizadores diferente do grupo de teste. Por este motivo, algumas interações $\boldsymbol{U} \times \boldsymbol{I}$ existentes no conjunto contendo todos os dados são aleatoriamente selecionadas e separadas para compor os dados de teste (os quais "fingimos não saber que existem") e o modelo utilizará todos os demais dados no treino. É importante que o subconjunto de teste obtido contemple, ao menos, uma recomendação existente de item a ser verificada para cada utilizador (método *user test*). Adicionalmente a utilização dos dados para a metodologia de treinamento e teste, podem ser alterados todos os parâmetros utilizados no modelo computacional (no exemplo concreto, a classe NCF) em busca de uma melhor assertividade na recomendação. Normalmente em RS a quantidade de épocas de treino do modelo tende a ser um número pequeno, ou seja, poucas épocas tornam-se suficientemente necessárias para se obter um bom ajuste de acurácia.

Deste modo, analisando-se os resultados obtidos por métricas de avaliação dos RS em execuções distintas de instâncias da classe NCF apresentada no Código-fonte 1, verifica-se na prática que a abordagem de filtragem colaborativa sofre interferência direta do problema de "arranque a frio", dada a necessidade de colaboração entre as avaliações dos itens por utilizadores para a geração da recomendação. A CF tende a obter um desempenho proporcional a quantidade de dados iniciais utilizada, ou seja, a utilização de poucos dados (pouca colaboração), resultará em menor assertividade e vice-versa. A falta de dados de avaliações é tratada melhor no modelo de filtragem baseada em conteúdo, observando-se as perdas encontradas no cálculo da similaridade entre os itens por diferentes equações (3-6), bem como pela utilização do cálculo da diversidade dado pela equação (7) e por outras medidas. A abordagem de filtragem baseada em conhecimento (KBF) tem sido menos utilizada em relação a CF e CBF, porém pode trazer ganhos em casos específicos e em modelos computacionais que implementam abordagens híbridas.

3.4. Metodologia de Avaliação da Recomendação

Um RS pode ser avaliado sob diferentes aspectos por meio de dados iniciais obtidos de forma *off-line* e *on-line*. As principais métricas utilizadas são baseadas em dados, permitem a produção de indicadores de qualidade e a comparação destes entre as diferentes heurísticas empregadas nos modelos computacionais, como por exemplo, as métricas de precisão (*precision*) e de lembrança (*recall*) da recomendação. Normalmente a mensuração é feita pela média da assertividade entre todos os utilizadores, avaliando numa escala percentual se os itens desejados (com interação: avaliação, visitação, consumo, ...) estão entre os *Top*-N itens recomendados para cada utilizador. A Figura 8 ilustra para cada situação verdadeira (positivo e negativo) ou falsa (positivo e negativo) ocorrida entre a recomendação gerada e o consumo/visitação pelo utilizador, como poderá ser montado um quadro do tipo teste A/B para ser computada uma matriz de confusão contendo a frequência da classificação obtida.

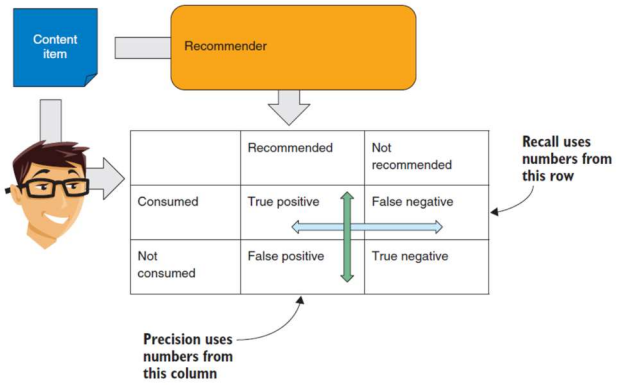


Figura 8. Formação do cálculo utilizado nas métricas *precision* (por colunas) e *recall* (por linhas) na relação entre itens recomendados e consumidos/visitados, retirada de Falk [2019, p. 226].

Considerando-se a interceptação entre linhas e colunas no quadro ilustrado na Figura 8, podem ser calculados alguns indicadores de qualidade, como por exemplo: *precision*, *recall*, *accuracy* e F-*score* ("situação do placar") formulados abaixo, entre outros:

- Qual a proporção de identificações positivas foi realmente correta? Considerando a Figura 8, qual a fração de itens recomendados o utilizador consumiu/visitou?

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (8)$$

- Qual a proporção de positivos foi identificada corretamente? Na Figura 8, entre todos os itens que o utilizador consumiu/visitou, quantos foram recomendados?

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (9)$$

- Quanto acertou? Na Figura 8, seria a proporção das previsões corretas em relação a todas as previsões:

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative} \quad (10)$$

- O F-*Score* mostra um balanço entre as métricas *precision* e *recall*:

$$\text{F-}Score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (11)$$

Em projetos tradicionais de ML, comumente os modelos implementados são avaliados utilizando métricas como acurácia (*accuracy*) para problemas de classificação e para problemas de regressão as métricas do erro absoluto médio (*Mean Absolute Error* – MAE) e erro quadrático médio da raiz (*Root Mean Square Error* – RMSE). Uma diferenciação é que o MAE considera o mesmo peso para qualquer erro, enquanto o RMSE eleva o erro ao quadrado antes de considerá-lo e tende a penalizar o tipo de erro [Maheswari et al. 2019]. Entretanto, essas métricas são consideradas simplistas para a avaliação da eficiência das recomendações e foram desenvolvidos, por meio de investigações no assunto, alguns protocolos de avaliações específicos. Basicamente, um protocolo de avaliação em RS procura comprovar se o utilizador

interagiu ou tende a interagir com, pelo menos, uma recomendação de item, encontrando um número numa escala percentual de 0 a 100%.

Normalmente as métricas@N (métricas em N) são calculadas pela média, acumulando todos os indicadores obtidos numa verificação de frequência, por exemplo utilizando as equações (8), (9) e (10), sendo dividido pela quantidade de utilizadores $|\boldsymbol{U}|$, em que N é igual ao número de itens recomendados e verificados entre todos os itens do conjunto $\boldsymbol{I}$. São exemplos de métricas de avaliação comumente utilizadas em RS: Taxa de acerto na posição – *Hit Ratio* (HR@N), Precisão (*Precision*@N), Lembrança (*Recall*@N), *Mean Average Precision* (MAP), *Normalized Discounted Cumulative Gain* (nDCG@N), *Coverage* (COV@N), *Average Popularity* (POP@N), que normalmente não consideram a ordem dos itens e *Mean Recipricol Rank* (MRR@N), que considera a ordem dos itens, entre outras [Ludewig e Jannach 2018; He et al. 2017; Shani e Gunawardana 2010].

Para avaliar os RS a comunidade científica tem utilizado predominantemente as métricas que buscam quantificar a precisão na recomendação, tais como: *Precision*@N e *Recall*@N, o ranqueamento médio MRR@N e/ou a taxa de acerto HR@N. São encontrados poucos trabalhos na literatura de referência que apresentam outras métricas de avaliação como a cobertura, a diversidade, a novidade ou ainda itens mais populares, estando este tópico em crescimento em investigações.

4. Recomendação Sequencial em Ambientes *On-line*

Entre as técnicas de inteligência artificial que podem aprimorar os RS estão as arquiteturas DNN e estudos recentes apontam em uma direção conhecida como recomendação sequencial (*sequential recommendation*) apoiada pela recomendação baseada em sessão (*session-based recommendation*) em ambientes *on-line* [Q. Zhang et al. 2021; Kim et al. 2021; Mi et al. 2020; Dokoohaki 2020]. Em sistemas de recomendação sequencial são considerados os dados sequenciais históricos obtidos em intervalos temporais para prover a recomendação. Desta forma, a recomendação pode ser melhorada continuamente por meio dos estados anteriores serem modelados por recorrência para prover os próximos *Top*-N itens de interesse dos utilizadores de forma individualizada.

Hidasi et al. [2016] caracterizam que nesta classe especializada de RS, além do preconizado objetivo de se obter recomendações relevantes de itens para utilizadores em tempo computacional não proibitivo, as recomendações sequenciais produzidas requerem adaptação contínua, pois necessitam considerar os utilizadores e os itens novos e obsoletos em interações recorrentes. Em ambientes *on-line* torna-se preciso considerar ainda a utilização de *feedbacks* implícitos, que podem ser gerados em larga escala de forma recorrente na navegação e que podem ser transformados em *feedbacks* explícitos. De acordo com Xu He et al. [2020, p. 9] "O motivo é que o ambiente *on-line* é mais complexo do que a configuração *off-line* em termos de número de itens candidatos e a origem dos cliques.". No exemplo apresentado pelos autores foi construído um simulador em que o número de itens candidatos recebeu um aumento de 10 para 2.000 para cada conjunto simulado no ambiente *on-line*.

A Figura 9 ilustra a estrutura básica arquitetural dos sistemas de recomendação sequenciais. Além dos elementos identificados na Figura 2, esta estrutura normalmente é modelada e

constituída contendo outros dois módulos essenciais: um extrator de *feedbacks* implícitos para capturar a sequência de ações históricas do utilizador e um preditor de saída do modelo computacional para recomendar itens e prever as próximas ações do utilizador.

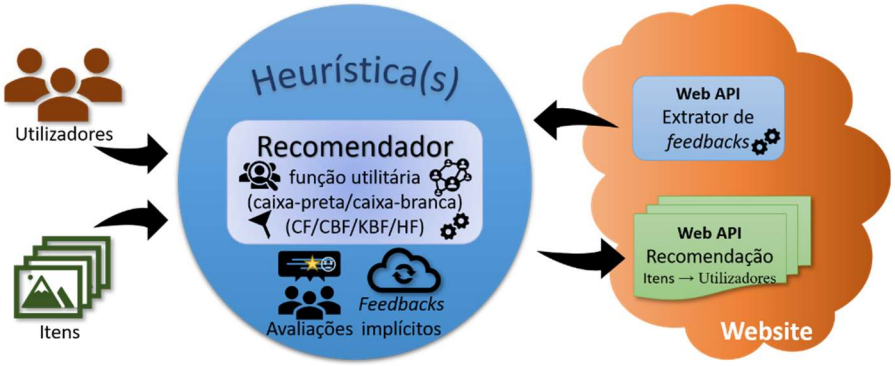


Figura 9. Estrutura básica empregada em sistemas de recomendação sequenciais em ambientes *on-line*.

A recomendação sequencial utiliza as informações ordenadas e pode ser implementada na prática dentro ou fora de sessões. Por sua vez, a recomendação baseada em sessão permite a utilização das informações de forma ordenada e também sem levar em consideração a sua ordenação sequencial, bem como pode considerar os dados providos em uma ou mais sessões [Wang et al. 2020]. Assim, a recomendação sequencial e baseada em sessão são complementares e promovem a construção de RS particularmente baseados em *feedbacks* implícitos e especialmente adaptativos em ambientes *on-line*, em que podem ser utilizados diversos recursos, como por exemplo o reconhecimento de padrões das preferências dos utilizadores para a personalização [Quadrana et al. 2017].

Modelos computacionais neuronais com mecanismos de autoatenção tem sido adotados em sistemas de recomendação sequenciais baseados em sessão [Mi et al. 2020; Kang e McAuley 2018]. Além disso, para obter êxito na recomendação adaptativa em ambientes *on-line*, os módulos estruturais devem ser projetados para atender a demanda para cada utilizador durante uma ou mais sessões monolíticas, requerendo ou não as credenciais de identificação dos utilizadores. Por fim, o conjunto de heurísticas a ser empregado deve considerar os cenários contendo dinâmicas sequenciais de curto prazo e de longo prazo.

5. Conclusão

Este artigo caracterizou os sistemas de recomendação apresentando um breve histórico, algumas definições importantes relacionadas ao tema, bem como uma taxonomia das suas principais abordagens. Também abordou algumas questões teórico-práticas vinculadas a representação e manipulação de dados e metadados, medidas usuais, algoritmos com DNN e metodologias de avaliação da recomendação de itens para utilizadores.

O crescimento em investigações ao longo das últimas três décadas resultou em um grande volume de produções bibliográficas vinculadas ao tema RS. Foi constatado o aumento na utilização do *deep learning* em investigações e uma tendência recente na recomendação sequencial apoiada pela recomendação baseada em sessão em ambientes *on-line*. Entretanto,

algumas verificações ainda trazem a dificuldade de uso e de generalização para diferentes domínios, tais como: múltiplos rastreamentos *on-line*, grandes volumes de dados heterogéneos e não interconectados e diferentes métodos matemáticos em uso nas tarefas de recomendação. Maior qualidade na recomendação é almejada com o emprego de recursos da inteligência artificial para melhorar a satisfação e a experiência do utilizador.

Os cenários e os desafios atuais constituem principalmente aspectos de produção e utilização dos RS a serem aprimorados com o estudo em heurísticas e em arquiteturas. É constatado que ao conhecer os diferentes tipos de métodos em ML, é possível mensurar que o DL é melhor empregado em problemas que especialistas humanos conseguem resolver e não em situações consideradas impossíveis para os seres humanos – não há mágica na recomendação. Um RS é constituído por vários componentes que realizam funções específicas ou conjunto de funções para resolver o problema de recomendação e as propriedades são tratadas por sua arquitetura, como a disponibilidade *on-line* vs. desempenho computacional, encapsulando definições, conceitos e funcionalidades que poderão ser reaproveitadas em outros projetos do mesmo tipo.

Por fim, além da caracterização apresentada neste artigo, para a utilização em escala dos RS tornam-se importantes as decisões por metodologias e recursos tecnológicos, a considerar as diferentes arquiteturas DNN e as definições paramétricas que possam ser empregadas para apoiar o projeto teórico e o projeto técnico. Também se torna importante o estudo para avaliar de forma mensurada os resultados obtidos nos modelos computacionais de recomendação, sobretudo em ambientes *on-line*, além de outras oportunidades que ainda se apresentarão para a investigação em RS.

Agradecimentos

Este trabalho foi apoiado pelo Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS) – Brasil.

Referências

ACM RecSys. (n.d.). *ACM RecSys Conferences Series on Recommender Systems*. https://recsys.acm.org. Acedido em 20/08/2021.

Adomavicius, G., & Tuzhilin, A. (2005). Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*, *17*(6), 734–749. https://doi.org/10.1109/TKDE.2005.99

Aggarwal, C. C. (2016). *Recommender Systems: The Textbook*. Springer International Publishing. https://doi.org/10.1007/978-3-319-29659-3

Castells, P., Vargas, S., & Wang, J. (2011). *Novelty and Diversity Metrics for Recommender Systems: Choice, Discovery and Relevance*. 8.

Dara, S., Chowdary, C. R., & Kumar, C. (2020). A survey on group recommender systems. *Journal of Intelligent Information Systems*, *54*(2), 271–295. https://doi.org/10.1007/s10844-018-0542-3

Dokoohaki, N. (Org.). (2020). Fashion Recommender Systems. In *Fashion Recommender Systems* (p. 3–21). Springer International Publishing. https://doi.org/10.1007/978-3-030-55218-3_1

Eberle, O., Büttner, J., Kräutli, F., Müller, K.-R., Valleriani, M., & Montavon, G. (2020). Building and Interpreting Deep Similarity Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1. https://doi.org/10.1109/TPAMI.2020.3020738

Ekstrand, M. D., Ludwig, M., Konstan, J. A., & Riedl, J. T. (2011). *Rethinking the Recommender Research Ecosystem: Reproducibility, Openness, and LensKit*. 8.

Falk, K. (2019). *Practical recommender systems*. Manning. Shelter Island, NY

Felfernig, A., Boratto, L., Stettinger, M., & Tkalčič, M. (2018). *Group Recommender Systems: An Introduction*. Springer International Publishing. https://doi.org/10.1007/978-3-319-75067-5

Friedrich, G., & Zanker, M. (2011). A Taxonomy for Generating Explanations in Recommender Systems. *AI Magazine*, *32*(3), 90–98. https://doi.org/10.1609/aimag.v32i3.2365

Gharahighehi, A., & Vens, C. (2021). Personalizing Diversity Versus Accuracy in Session-Based Recommender Systems. *SN Computer Science*, *2*(1), 39. https://doi.org/10.1007/s42979-020-00399-2

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, *35*(12), 61–70. https://doi.org/10.1145/138859.138867

Greenspan, H., San José Estépar, R., Niessen, W. J., Siegel, E., & Nielsen, M. (2020). Position paper on COVID-19 imaging and AI: From the clinical needs and technological challenges to initial AI solutions at the lab and national level towards a new era for AI in healthcare. *Medical Image Analysis*, *66*, 101800. https://doi.org/10.1016/j.media.2020.101800

Gunawardana, A., & Shani, G. (2009). *A Survey of Accuracy Evaluation Metrics of Recommendation Tasks*. 28.

He, X., An, B., Li, Y., Chen, H., Wang, R., Wang, X., Yu, R., Li, X., & Wang, Z. (2020). Learning to Collaborate in Multi-Module Recommendation via Multi-Agent Reinforcement Learning without Communication. *ArXiv:2008.09369 [Cs]*. http://arxiv.org/abs/2008.09369

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T.-S. (2017). Neural Collaborative Filtering. *arXiv:1708.05031 [cs]*. http://arxiv.org/abs/1708.05031

Herlocker, J. L., Konstan, J. A., & Riedl, J. (2000). Explaining Collaborative Filtering Recommendations. *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, 241–250. https://doi.org/10.1145/358916.358995

Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating Collaborative Filtering Recommender Systems. *ACM Trans. Inf. Syst.*, *22*(1), 5–53. https://doi.org/10.1145/963770.963772

Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2016). Session-based Recommendations with Recurrent Neural Networks. *ArXiv:1511.06939 [Cs]*. https://www.arxiv-vanity.com/papers/1511.06939/. http://arxiv.org/abs/1511.06939

Hiriyannaiah, S., Siddesh, G. M., & Srinivasa, K. G. (2020). Deep visual ensemble similarity (DVESM) approach for visually aware recommendation and search in smart community. *Journal of King Saud University - Computer and Information Sciences*, S1319157820303293. https://doi.org/10.1016/j.jksuci.2020.03.009

Jannach, D., Mobasher, B., & Berkovsky, S. (2020). Research directions in session-based and sequential recommendation. *User Modeling and User-Adapted Interaction*, *30*(4), 609–616. https://doi.org/10.1007/s11257-020-09274-4

Kang, W.-C., & McAuley, J. (2018). Self-Attentive Sequential Recommendation. *2018 IEEE International Conference on Data Mining (ICDM)*, 197–206. https://doi.org/10.1109/ICDM.2018.00035

Karlgren, J. (1990). *An Algebra for Recommendations* (Working Paper Nº 179; p. 11). KTH Royal Institute of Technology and Stockholm University. https://jussikarlgren.files.wordpress.com/1990/09/algebrawp.pdf

Karlgren, J. (1994). *Newsgroup Clustering Based On User Behavior-A Recommendation Algebra*. 15.

Kim, J., Wi, J., & Kim, Y. (2021, fevereiro). Sequential Recommendations on GitHub Repository. *Journal Applied Sciences*, 14.

Koren, Y. (2009). *The BellKor Solution to the Netflix Grand Prize*. Netflix prize documentation. http://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor. pdf. Acedido em 20/08/2021.

Levandoski, J. J., Ekstrand, M. D., Ludwig, M. J., Eldawy, A., Mokbel, M. F., & Riedl, J. T. (2011). RecBench: Benchmarks for Evaluating Performance of Recommender System Architectures. *Proc. VLDB Endow.*, *4*(11), 911–920. https://doi.org/10.14778/3402707.3402729

Li, J., Ren, P., Chen, Z., Ren, Z., Lian, T., & Ma, J. (2017). Neural Attentive Session-Based Recommendation. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 1419–1428. https://doi.org/10.1145/3132847.3132926

Liu, H., Hu, Z., Mian, A., Tian, H., & Zhu, X. (2014). A New User Similarity Model to Improve the Accuracy of Collaborative Filtering. *Knowledge-Based Systems*, *56*, 156–166. https://doi.org/10.1016/j.knosys.2013.11.006

Loy, J. (2020). *Deep Learning based Recommender Systems*. https://kaggle.com/jamesloy/deep-learning-based-recommender-systems. Acedido em 20/08/2021.

Ludewig, M., & Jannach, D. (2018). Evaluation of Session-Based Recommendation Algorithms. *User Modeling and User-Adapted Interaction*, *28*(4), 331–390. https://doi.org/10.1007/s11257-018-9209-6

Maheswari, M., Geetha, S., & Selva kumar, S. (2019). Adaptable and proficient Hellinger Coefficient Based Collaborative Filtering for recommendation system. *Cluster Computing*, *22*(5), 12325–12338. https://doi.org/10.1007/s10586-017-1616-7

Malone, T. W., Grant, K. R., Turbak, F. A., Brobst, S. A., & Cohen, M. D. (1987). Intelligent Information-Sharing Systems. *Commun. ACM*, *30*(5), 390–402. https://doi.org/10.1145/22899.22903

Mi, F., Lin, X., & Faltings, B. (2020). ADER: Adaptively Distilled Exemplar Replay Towards Continual Learning for Session-based Recommendation. *Fourteenth ACM Conference on Recommender Systems*, 408–413. https://doi.org/10.1145/3383313.3412218

Morais, A. J., Oliveira, E., & Jorge, A. M. (2012). A Multi-Agent Recommender System. In S. Omatu, J. F. De Paz Santana, S. R. González, J. M. Molina, A. M. Bernardos, & J. M. C. Rodríguez (Orgs.), *Distributed Computing and Artificial Intelligence* (Vol. 151, p. 281–288). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-28765-7_33

Quadrana, M., Karatzoglou, A., Hidasi, B., & Cremonesi, P. (2017). Personalizing Session-Based Recommendations with Hierarchical Recurrent Neural Networks. *Proceedings Eleventh ACM Conference on Recommender Systems*, 130–137. https://doi.org/10.1145/3109859.3109896

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). GroupLens: An Open Architecture for Collaborative Filtering of Netnews. *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, 175–186. https://doi.org/10.1145/192844.192905

Ricci, F., Rokach, L., & Shapira, B. (Orgs.). (2015). *Recommender Systems Handbook*. Springer US; DOI 10.1007/978-1-4899-7637-6. https://www.springer.com/gp/book/9781489976369

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, *24*(5), 513–523. https://doi.org/10.1016/0306-4573(88)90021-0

Salton, G., & McGill, M. J. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill. https://vpn.uab.pt/https/dl.acm.org/doi/book/10.5555/576628

Shani, G., & Gunawardana, A. (2010). Evaluating Recommendation Systems. In F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor (Orgs.), *Recommender Systems Handbook* (p. 257–297). Springer. https://doi.org/10.1007/978-0-387-85820-3_8

Shao, B., Li, X., & Bian, G. (2021). A survey of research hotspots and frontier trends of recommendation systems from the perspective of knowledge graph. *Expert Systems with Applications*, *165*, 113764. https://doi.org/10.1016/j.eswa.2020.113764

Skansi, S. (2018). *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*. Springer International Publishing. https://doi.org/10.1007/978-3-319-73004-2

Smyth, B., & McClave, P. (2001). Similarity vs. Diversity. *Case-Based Reasoning Research and Development*, 347–361. https://doi.org/10.1007/3-540-44593-5_25

Taghavi, M., Bentahar, J., Bakhtiyari, K., & Hanachi, C. (2018). New Insights Towards Developing Recommender Systems. *The Computer Journal*, *61*(3), 319–348. https://doi.org/10.1093/comjnl/bxx056

Wang, S., Cao, L., Wang, Y., Sheng, Q. Z., Orgun, M., & Lian, D. (2020). A Survey on Session-based Recommender Systems. *arXiv:1902.04864 [cs]*. http://arxiv.org/abs/1902.04864

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2021). *Dive into Deep Learning*. http://d2l.ai/

Zhang, Q., Lu, J., & Jin, Y. (2021). Artificial intelligence in recommender systems. *Complex & Intelligent Systems*, *7*(1), 439–457. https://doi.org/10.1007/s40747-020-00212-w

Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). Deep Learning based Recommender System: A Survey and New Perspectives. *ACM Computing Surveys*, *52*(1), 1–38. https://doi.acm.org/10.1145/3285029

Rogério Xavier de Azambuja, Professor de Engenharia de Software no Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS) Campus Farroupilha, Farroupilha/RS – Brasil. Mestre em Ciência da Computação (UFSC – Brasil). Bacharel em Informática (UNIJUÍ – Brasil). Licenciado em Informática (IFRS – Brasil). Trabalho produzido nos Ensaios de Aprofundamento do Curso de Doutoramento em Ciência e Tecnologia Web (UTAD/UAb – Portugal). Interesse na investigação em Engenharia de Software (*Software Engineering*) e Sistemas de Recomendação (*Recommender Systems*).

Jorge Moraes, Professor Auxiliar no Departamento de Ciências e Tecnologia da Universidade Aberta (UAb). Presidente do Conselho Pedagógico. Investigador do Laboratório de Inteligência Artificial e Apoio à Decisão (LIAAD – INESC TEC L. A.).

Vitor Filipe, Professor Associado com Agregação da Escola de Ciências e Tecnologias da Universidade de Trás-os-Montes e Alto Douro (UTAD), Portugal. Doutoramento em Engenharia Electrotécnica pela UTAD, em 2003 e mestrado em Informática obtido na Universidade do Minho, Portugal. Investigador sénior do INESC TEC onde desenvolve trabalho nas áreas de Visão por Computador e Aprendizagem Automática aplicadas em diferentes domínios.

(esta página par está propositadamente em branco)