MATHEMATICAL MODELING AND COMPUTING, Vol. 11, No. 4, pp. 1078–1092 (2024)

Recommendation systems techniques based on generative models and matrix factorization: a survey

Filali Zegzouti S.[1], Banouar O.[2], Benslimane M.[1]

[1] *Sciences, Engineering and Management Laboratory, Sidi Mohamed Ben Abdellah University, Fez, Morocco*
[2] *Laboratory of Computer Science Engineering and Systems, Cady Ayyad University, Marrakesh, Morocco*

(Received 20 January 2024; Revised 19 August 2024; Accepted 22 August 2024)

Collaborative filtering (CF) is a technique that can filter out items that a user might like based on the behaviors and preferences of similar users. It is a key en-abler technique for an effective recommendation system (RS). Model-based recommendation systems, a subset of CF, use data, typically ratings, to construct models for providing personalized suggestions to users. Our objective in this work is to provide a comprehensive overview of various techniques employed in Model-based RS, focusing on their theoretical foundations and practical applications. We explore the core challenges associated with recommendation, including the top-N recommendation problem, and explore the state-of-the-art model-based methods used to address these challenges. In this survey, we categorize these techniques into three distinct classes: matrix factorization, similarity-based, and completion-based methods. To compare their performance, we evaluated these techniques over the MovieLens datasets using two metrics: Mean Average Precision (MAP), Normalized Discounted Cumulative Gain (NDCG), precision and recall.

Keywords: *recommendation systems; model-based; top-N recommendation; generative adversarial network; matrix factorization; matrix completion.*

2010 MSC: 68Txx **DOI:** 10.23939/mmc2024.04.1078

1. Introduction

In today's era of digital abundance, where users are inundated with a wealth of in-formation across various platforms, recommendation systems (RS) have emerged as indispensable tools for curating personalized content experiences [1, 2]. Rooted in data-driven methodologies, these systems have transformed the way users discover tailored products, content, and services. RS are essential for providing users with tailored suggestions based on data [1–3]. Typically, these data comprise historical user profiles, encompassing records of a user's past interactions with items, and item/user metadata, which contain individual items and user information such as movie actors, song loudness, or demographic data [4]. User feedback, whether explicit or implicit [5], plays a pivotal role in RS. Explicit feedback involves clear indications of a user's interest in an item, often expressed through ratings or reviews, and is commonly employed by platforms such as Netflix and Amazon [3]. In contrast, implicit feedback is gathered indirectly during user interactions with the RS, such as selecting and watching a movie on a video-on-demand service, indicating some level of interest. However, implicit feedback's unary nature raises challenges concerning negative experiences. RS can be constructed using three main approaches [6]: content-based filtering [7, 8], which recommends items such as those in a user's historical profile; collaborative filtering [9–11], which identifies similar users and recommends items they have rated; and hybrid systems [8, 12], which combine collaborative and content-based filtering for more robust recommendations. In this paper, we have focused our review on model-based methods, which are subset of CF techniques that leverage various approaches to enhance recommendation systems. Among model-based methods, we delve into the core techniques of Matrix Factorization (MF) [4, 13–15], Matrix Completion using quantum computing [16, 17], and similarity-based approaches [18–20]. These approaches collectively represent the backbone of model-based collaborative filtering techniques. Initially, we compiled and scrutinized leading articles and prestigious conference publications to ensure a

dependable evaluation of the recommendation system models within these approaches. Subsequently, we structured studies pertaining to model-based recommendation system models and the underlying technology, subsequently exploring discernible research trajectories. Moreover, we categorized these approaches into three classes, delving into the recommendation models and techniques adopted within each class. Thus, this study not only offers a comprehensive overview of model-based RS, but also offers valuable insights to researchers interested in the diverse technologies and evolving trends within the application domains of Deep Learning-based generative models used for recommendation systems. The remaining of this paper is organized as the following. Section 2 presents our classification of model-based approaches for Recommendation Systems. Section 3, 4, 5, details namely Generative model-based Matrix Factorization (MF), similarity-based models, and matrix completion-based models, respectively. In Section 6, we present and discuss the obtained results. The Conclusion closes the paper.

2. Model based approaches for recommendation systems

Collaborative filtering [9–11, 21, 22] can be classified into two categories: memory-based and model-based approaches [6, 11, 22]. They are used in various fields, including machine learning and recommendation systems. In the context of recommendation systems, while memory-based [23] methods make recommendations based on user similarity or item similarity without explicitly creating a model, model-based methods create predictive models that represent the underlying relationships and patterns in input data. Model-based recommendation systems use mathematical or statistical functions in order to capture complex pat-terns and latent factors from the data. To achieve this, they use a variety of techniques (see Figure 1), such as Bayesian models [24, 25], clustering algorithm [12], and MF. Some common techniques used in MF includes SVD (Singular Value Decomposition) [12, 26, 27], NMF (Non-negative Matrix Factorization) [28, 29], ALS (Alternating Least Squares) [30], PCA (Principal component analysis) [31], and PMF (Probabilistic Matrix Factorization) [32, 33].

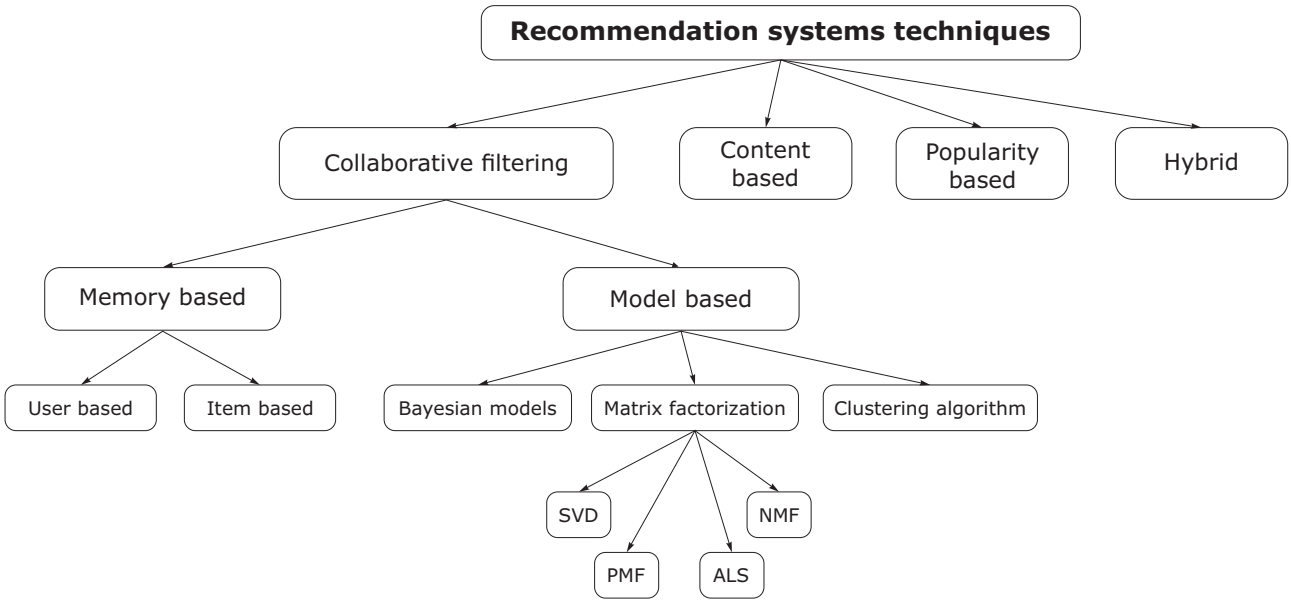

Fig. 1. Some recommendation systems techniques.

These algorithms play a crucial role in data analysis and machine learning, enabling the extraction of valuable insights and predictions from complex datasets. Model-based algorithms typically involve a learning process where the algorithm adjusts the model's parameters to fit the observed data. This process often includes optimization techniques to find the best-fitting model. Model-based collaborative filtering approaches, though effective in many cases, face key challenges. One is the "Cold Start issue", where they struggle with new users or items lacking enough data. Also, as datasets grow, scalability

becomes an issue, leading to higher computational demands. Overfitting is a concern, as models might become overly specialized to the training data, potentially compromising their ability to generalize well to diverse user preferences. Data sparsity poses another obstacle, especially in scenarios where users rate only a few items. Additionally, some complex models lack interpretability, making it hard to understand their recommendations. In this survey, we classify these techniques into three distinct classes:

— Generative models-based Matrix Factorization (MF),
— Similarity-based methods,
— Matrix completion-based methods

In the upcoming, we will delve deeply into algorithms recently employed within each category. Subsequently, we will conduct a quantitative comparison after applying these algorithms to two MovieLens datasets (1M ML and HetRec).

3. Generative models-based matrix factorization

It refers to an approach in RS that combines matrix factorization techniques with generative models. On the one hand, matrix factorization algorithms [13–15,34,35], in the case of CF, work by decomposing the user-item rating matrix (URM) into the product of two lower dimensionality rectangular matrices, which are latent factors for users and items. In the context of RS, the goal is to predict missing ratings or recommend new items to users.

On the other hand, generative models are class of models that are designed to generate new data samples that resemble a given training dataset. In the context of RS, they can be applied to capture complex relationships and patterns in user-item inter-actions. In this paper, our primary focus centers on a specialized class of deep learning models known as Generative Adversarial Networks, or GANs for short [36–38]. These groundbreaking models, first conceived by Goodfellow et al. in 2014 [36], have rapidly been expanding into a variety of fields, including computer vision, natural language processing, speech processing, and even cyber-security applications like mal-ware detection. GANs have proven their versatility and effectiveness in generating and manipulating data, making them indispensable tools across a wide spectrum of fields and industries. Mathematically: the original GAN is defined as a two-player game (see Figure 2) involving the following elements:

— Probability Space: denoted as $(\Omega, \mu_{\text{ref}})$, where Ω represents the sample space, and μ_{ref} represents a probability measure on that space. This defines the GAN game.
— Two Players: the game involves two players, namely the generator and the discriminator.
— Generator's strategy set: the generator's strategy set is represented as $\mathbb{P}(\Omega)$, which is the set of all probability measures μ_G on the sample space Ω.
— Discriminator's strategy set: the discriminator's strategy set consists of Markov kernels $\mu_D \colon \Omega \to \mathbb{P}([0,1])$, where $\mathbb{P}([0,1])$ is the set of probability measures on the interval $[0,1]$.
— Zero-sum game: the GAN game is a zero-sum game, meaning that the gain of one player is exactly balanced by the loss of the other player.
— Objective function: typically includes terms related to both the discriminator's ability to distinguish between real and generated items and the generator's ability to generate realistic items (Loss Function (GAN)=Loss Function (G)+Loss Function (D)+Regularization Part).

The objective function of the GAN game is defined as follows:

$$L\left(\mu_G, \mu_D\right) = E\left[x \sim \mu_{\text{ref}}, y \sim \mu_D(x)\right]\left[\ln(y)\right] + E\left[x \sim \mu_G, y \sim \mu_D(x)\right]\left[\ln(1-y)\right].$$

Here, E denotes the expectation operator, $x \sim \mu_{\text{ref}}$ signifies sampling from the reference distribution, and $x \sim \mu_{\text{ref}}$ represents sampling from the distribution produced by the discriminator given input x. The objective consists of two terms, one involving the reference distribution and the other involving the generator's distribution. The goal of the generator is to minimize this objective function, while the goal of the discriminator is to maximize it. This adversarial process drives the generator to produce data that is indistinguishable from real data, ultimately leading to improved generative capabilities.

In the original paper and most subsequent papers, it is typically assumed that the generator makes the first move, followed by the discriminator, resulting in the following minimax game formulation:

$$\min_{\mu_G} \max_{\mu_D} L(\mu_G, \mu_D) = E[x \sim \mu_{ref}, y \sim \mu_D(x)][\ln(y)] + E[x \sim \mu_G, y \sim \mu_D(x)][\ln(1-y)].$$

In this formulation, the generator aims to minimize the objective function, and the discriminator aims to maximize it.

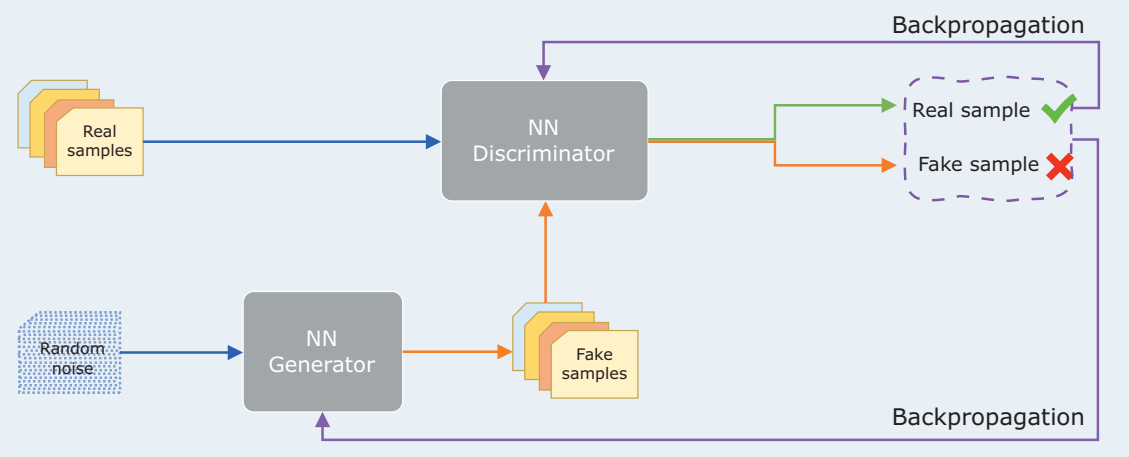

Fig. 2. Architecture of GAN.

In this section, we focus on two models-based Matrix factorization (PureSVD, and WRMF), three models-based variants of generative models (CFGAN, CAAE, and GraphGAN), and one model that combine both Matrix Factorization techniques with Generative Models, called GANMF.

3.1. PureSVD

PureSVD, introduced by Cremonesi et al. in 2009 for solving the Top-N recommendation problem, utilizes Singular Value Decomposition (SVD) to factorize the UserItem Rating Matrix (URM). SVD decomposes the URM into three matrices: M, a real orthonormal matrix of dimensions $|U| \times K$ representing user latent factors; Σ, a $K \times K$ matrix containing only the top K singular values; and N, a real orthonormal matrix of dimensions $|I| \times K$ representing item latent factors. This decomposition enables the extraction of user and item latent factors. In the application of SVD, missing ratings in the URM are typically imputed with zeros, although the authors note that other imputation values do not significantly affect the algorithm's performance. PureSVD is considered one of the top-performing baseline models for the Top-N recommendation problem [12,18,26,27]. However, it has some computational limitations, as it performs the decomposition into dense matrices even when the URM is sparse, rendering optimized libraries for sparse matrices unusable and potentially causing memory issues.

3.2. Weighted Regularized Matrix Factorization (WRMF)

Weighted Regularized Matrix Factorization (WRMF) is a Matrix Factorization technique tailored for implicit feedback in recommendation systems. It employs alternatingleast-squares (ALS) [30] as its training approach [14]. The process involves converting the User-Item Rating Matrix into a binary matrix, with 1 denoting user-item interaction and 0 denoting no interaction. WRMF assigns a confidence value (c_{ui}) to each interaction, representing the system's confidence in the user's preference for an item. This confidence value is calculated using a formula involving a factor α and the binary interaction. WRMF aims to minimize an optimization function that incorporates confidence-weighted interactions, user and item latent factors, and regularization terms to prevent overfitting. The model is trained iteratively using ALS, alternating between fixing user or item latent factors while optimizing the other. When fixing item latent factors, WRMF derives closed-form solutions for user latent factors based on confidence-weighted interactions. Similarly, when fixing user latent factors, it computes closed-form solutions for item latent factors using confidence-weighted interactions [39].

3.3. Conditional Feedback Generative Adversarial Network (CFGAN)

Conditional Feedback Generative Adversarial Network (CFGAN) [21] represents an innovative approach to the integration of Generative Adversarial Networks (GANs) into the realm of RS, with a specific emphasis on tackling the Top-N recommendation challenge. The developers of CFGAN have meticulously identified limitations in the training methodology of another GAN-based recommendation model called IRGAN, and they have documented their claims with rigorous experimental evidence in [34]. In response to these identified issues, CFGAN introduces a pioneering technique known as "vector-wise training" tailored specifically for GANs operating within the RS domain. The CFGAN minimax objective is given as:

$$J^D = -\sum_u \left(\log(r_u|c_u) + \log\left(1 - D\big((\widehat{r}_u \odot e_u)|c\big)\right)\right),$$

$$J^G = -\sum_u \log\left(1 - D\big((\widehat{r}_u \odot e_u)|c\big)\right).$$

In the context described, we have objectives J^D and J^G being minimized by D and G, respectively, and we are dealing with generated profiles $\hat{r}$ for user u, user conditioning vectors c_u, and masking vectors e_u defined as $e_u = r_u$ with element-wise multiplication represented by the symbol $\odot$.

This approach revolves around treating a user's past interactions as a structured vector. During training, the GAN's generator is meticulously taught to create realistic and deterministic historical user profiles. This stands in contrast to the conventional approach of randomly selecting relevant items. In this paradigm, the discriminator's primary task is to distinguish between GAN-generated user profiles and authentic ones. CFGAN follows a conditional GAN (cGAN) framework. Here, the generator network (G) receives a user conditioning vector (c_u), an unique user identifier and crucial input. Notably, CFGAN differs from typical cGANs by omitting the incorporation of random noise vectors (z). This omission is intentional, as CFGAN's main goal is to generate deterministic user profiles. These generated profiles are seamlessly merged with real ones for training the discriminator (D). As a result, this training approach enhances the discriminator's ability to classify GAN-generated profiles, leading to more effective and accurate performance.

3.4. Collaborative Adversarial Autoencoders (CAAE)

CAAE consists of three key models: the positive item generator (referred to as G), the negative item generator (referred to as G_0), and the discriminator (referred to as D) [22]. The roles of G and D in our framework are akin to those of G and D in other Generative Adversarial Networks (GANs). Specifically, when given a user, G primary objective is to generate a set of items that the user may find interesting and potentially choose. Meanwhile, D role is to differentiate between these generated items and the user's actual items derived from ground truth data. Furthermore, we introduce another generative model called G_0, which is dedicated to generating negative items-items that the user is unlikely to be interested in. This G_0 model is seamlessly integrated into this framework [38].

Discriminator: formally, the objective function of D is $J^D = l^D_{GAN} + \beta \cdot \text{reg}(\varphi)$.

J^D comprises two key elements: the GAN loss (L^D_{GAN}) and $L2$ regularization ($\text{reg}(\varphi)$), controlled by β. It emphasizes:

- J^D components: J^D combines L^D_{GAN} and $\text{reg}(\varphi)$, representing GAN loss and $L2$ regularization, respectively.
- Model parameters: φ signifies $D's$ model parameters.
- $L2$ regularization: $\text{reg}(\varphi)$ prevents overfitting, computed as half of $\varphi's$ $L2$ norm.
- Binary classification: D classifies "real" user items from ground truth vs. "fake" items generated by G or G_0.
- Preference learning: BPR-MF is used for learning relative user item preferences, improving over standard MF.

— Pairwise preferences: D is trained via a pairwise objective, maximizing the likelihood of user preferences to distinguish "real" from "fake" items.
— Discrimination function: $D(i,j|u)$ calculates the probability of item i being preferred over j for user u based on item characteristics.
— Overall objective: J^D combines expectations over user item pairs, weighted by the log of the discrimination function output and $L2$ regularization.

Positive item generator: The objective function of the positive element generator (G), denoted as J^G, $j^G = \lambda \cdot l^G_{\text{GAN}} + (1-\lambda) l_{\text{AE}} + \beta \cdot \text{reg}(\theta)$ comprises three crucial elements:

— GAN loss ($\lambda * L^G_{GAN}$): this part represents the loss associated with the Generative Adversarial Network (GAN) and is controlled by the parameter λ to maintain balance.
— Autoencoder loss ($(1-\lambda) * L_{AE}$): it quantifies the loss associated with the Autoencoder (AE) within the model, which focuses on reconstructing a dense purchase vector from a sparse one.
— $L2$ regularization ($\beta \cdot \text{reg}(\theta)$): this element introduces $L2$ regularization, regulated by the parameter β, to prevent overfitting.

The Autoencoder serves as a neural network tasked with reconstructing input data by learning hidden patterns. In the context of collaborative filtering (CF), it takes a sparse purchase vector as input and generates a denser vector with predicted values for items not yet purchased. The Autoencoder architecture includes a hidden layer with weight matrices and biases. The Autoencoder loss function ($L_{AE} = \sum_u \| (r_u - \hat{r}_u) \cdot e_u \|$) employs squared error terms and an indicator vector (e_u) to account for missing entries in the purchase vector. A softmax function is utilized to generate plausible items for users based on the Autoencoder's output, facilitating the computation of item selection probabilities. The GAN loss of $G(L^G_{GAN})$ leverages this probability distribution. The overall objective function for $G(J^G)$ combines the components of GAN loss, Autoencoder loss, and $L2$ regularization.

Negative Item Generator: the Negative Item Generator, G_0, shares similarities with G but has a distinct role. Like G, G_0 employs an Autoencoder to calculate softmax probabilities for item generation. However, G_0 primary aim is to generate negative items for a given user, this expressed $(p_{\theta'}(i_k|u) = \frac{\exp(\hat{r}_{i_k} u)}{\sum_i \exp(\hat{r}_{iu})})$, where $p_{\theta'}(i_k|u)$ represents the probability of selecting item i_k as a negative choice for user u. In contrast to G, G_0 has an opposing objective. While G aims to deceive the discriminator D, G_0 assists D in correctly distinguishing between real and fake items from G_0. The objective function for G_0, denoted as $(J_{G_0} = -E_{i \sim p_{\text{true}}(i|u), j \sim p_{\theta'}(j|u)} [\ln D(i,j|u)] + \frac{\beta}{2} \cdot |\theta'|^2)$, differs from that of G. Notably, it excludes the reconstruction loss function.

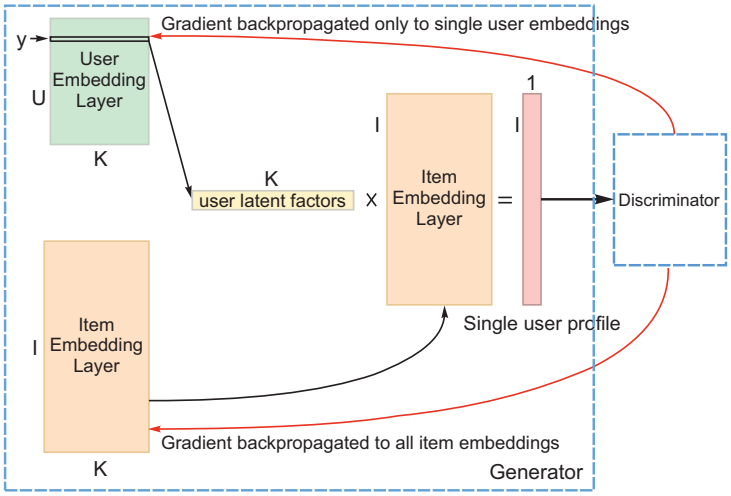

Fig. 3. Overview of our CAAE framework [33].

3.5. Learning graph representations using generative adversarial networks (GraphGAN)

In GraphGAN framework, the central focus lies in the careful selection of appropriate generator and discriminator components for the task of graph representation learning [40]. The traditional softmax functions, however, prove inadequate for the role of the generator for two fundamental reasons:

— Absence of Graph Structure Awareness: conventional softmax treats all vertices of a graph uniformly when considering a specific vertex, neglecting crucial information relating to the graph's inherent structure and proximity.

— Computational inefficiency: the computation of softmax entails the involvement of all vertices within the graph, resulting in a computationally demanding and inefficient process.

To surmount these challenges, the authors introduce an innovative generator concept named "Graph Softmax" in [40]. This approach reimagines how connectivity distributions are defined within a graph and boasts advantageous traits such as normalization, a heightened understanding of the graph's structure, and enhanced computational efficiency. Furthermore, a novel online generating strategy is proposed for the generator, grounded in random walks, and aligns seamlessly with the principles underpinning graph softmax. This strategic shift significantly reduces the computational complexity associated with the process. The overarching framework of GraphGAN is articulated as follows: When provided with a graph $G = (V, E)$, where $V = v_1, \ldots, v_N$ signifies vertices and $E = \{e_{ij}\}_{i,j \in V}$ denotes edges. We define "$N(v_c)$" as the set of vertices directly connected to a specific vertex "v_c". We use the term "conditional probability", referred to as "$p_{\text{true}}(v|v_c)$", to describe the true connectivity distribution for vertex "v_c". This probability helps us understand the preferences of "v_c" when it comes to connecting with other vertices within the larger set "V". We can view "$N(v_c)$" as a collection of observed samples that are drawn from this "$p_{\text{true}}(v|v_c)$" distribution. The primary objective is to cultivate two distinct models:

— Generator (G): this model ($G(v|v_c; \theta_G)$) strives to approximate the authentic connectivity distribution for a given vertex v_c, either selecting or generating the most probable vertices for connection with v_c from the set of vertices, V.
— Discriminator (D): in contrast, the discriminator, $D(v, v_c; \theta_D)$, is tasked with discerning the connectivity status between pairs of vertices (v, v_c) and produces a scalar indicating the likelihood of an edge existing between them.

In this dynamic interplay, the generator and discriminator engage in a two-player minimax game. G endeavors to meticulously mimic $p_{\text{true}}(v|v_c)$ and generate pertinent vertices, aiming to deceive D. Conversely, D endeavors to distinguish between genuine neighbors of v_c and those conjured by G. The objective functions for both G and D are meticulously defined, invoking a minimization of θ_G for G and a maximization of θ_D for D. The essence of their competition is encapsulated within the value function $V(G, D)$. In each iterative cycle, D is primed with positive and negative samples, whereas G is iteratively refined using policy gradients under the astute guidance of D. This ongoing competitive dynamic persists until G attains an indistinguishable likeness to the genuine connectivity distribution. Formally, G and D are playing the following two-player minimax game with value function:

$$\min_{\theta_G} \max_{\theta_D} V(G, D) = \sum_{c=1}^{N} \left(E_{v \sim p_{\text{true}}(\cdot|v_c)} \left[\log D(v, v_c; \theta_D) \right] + E_{v \sim G(\cdot|v_c; \theta_G)} \left[1 - \log D\left(v, v_c; \theta_D\right) \right] \right).$$

3.6. GAN-based Matrix Factorization for Recommendation Systems (GANMF)

Generative Adversarial Network with Matrix Factorization is an advanced machine learning model tailored for user profile generation [15]. This model seamlessly integrates Matrix Factorization techniques into the framework of a Conditional Generative Adversarial Network (cGAN). Here is a concise overview of GANMF:

— Conditional GAN (cGAN): GANMF is firmly grounded in the concept of a Conditional Generative Adversarial Network (cGAN) [14]. In this setup, the generator's actions are guided by specific information, represented as a user identifier 'y' in GANMF. The objective function in a cGAN is:

$$\min_G \max_D E_{x \sim p_{\text{data}}}[\log D(x|c)] + E_{z \sim p_z}[\log(1 - D)].$$

— Autoencoder-Based Discriminator: in contrast to traditional cGANs that employ binary classifier discriminators, GANMF adopts an autoencoder-based discriminator. This unique discriminator takes user profiles, whether real or synthetic, and produces a reconstruction of the input. It utilizes the reconstruction error to differentiate genuine from generated profiles. The reconstruction error of the autoencoder acts as the energy function: $D(x) = \| \operatorname{Dec}(\operatorname{Enc}(x)) - x \|_2^2$.

— Matrix Factorization (MF): GANMF combines GAN architecture with Matrix Factorization techniques innovatively. It includes two embedding layers: one for encoding user latent factors and another for item latent factors. These latent factors capture crucial data patterns. In generating synthetic user profiles, GANMF utilizes these latent factors along with the user identifier 'y' to produce profiles that reflect individual user preferences and behaviors.

— Profile Generation: in the generation process, the 'y' value plays a central role. It is used to identify the specific row in the user latent factor matrix corresponding to the given user. By accessing this row, the generator obtains vital information to craft a synthetic user profile that accurately represents the user associated with 'y'. A synthetic user profile is generated as follows (see Figure 4): $G(y) = V\Sigma[y, :]\top$.
To fool the discriminator, the generator minimizes the reconstruction error of the discriminator on generated profiles: $L^G(y) = D(G(y)) + \lambda_G * \|\Omega_G\|_2^2$, where '$D$' represents the discriminator function, 'G' represents the generator function, 'y' denotes the user row within the URM, λ_G is the regularization coefficient for L_2 regularization, and Ω_G includes the parameters associated with the generator.

— Loss Functions: GANMF employs specialized loss functions for effective training of both the generator and the discriminator. The generator's main goal is to minimize the reconstruction error assessed by the discriminator when evaluating the generated profiles. Furthermore, GANMF introduces additional loss terms, such as maximum mean discrepancy, to strike a balance between ensuring the plausibility of generated profiles and their ability to capture individual user characteristics. The parameter 'α' in these loss terms allows for fine-tuning of this trade-off. Therefore, we modify L^G by adding this additional loss:

$$L^G(x, y) = (1 - \alpha)D(G(y)) + \alpha\| \operatorname{Enc}(x) - \operatorname{Enn}(G(y))\| + \lambda_G\|\Omega_G\|_2^2.$$

In this scenario, 'Enc' represents the output produced by the encoder within the GANMF discriminator. The variable 'y' corresponds to the user conditioning attribute, while 'x' stands for the real user profile associated with 'y'. Additionally, 'α' serves as a constant that regulates the trade-off between the adversarial and feature matching losses.

— Training: GANMF's training regimen follows an iterative process, with alternating updates to the generator and discriminator. This iterative training continues until the discriminator can no longer distinguish between synthetic profiles generated by the model and real profiles from the dataset. The goal is to train the generator to produce profiles that are nearly indistinguishable from genuine ones.

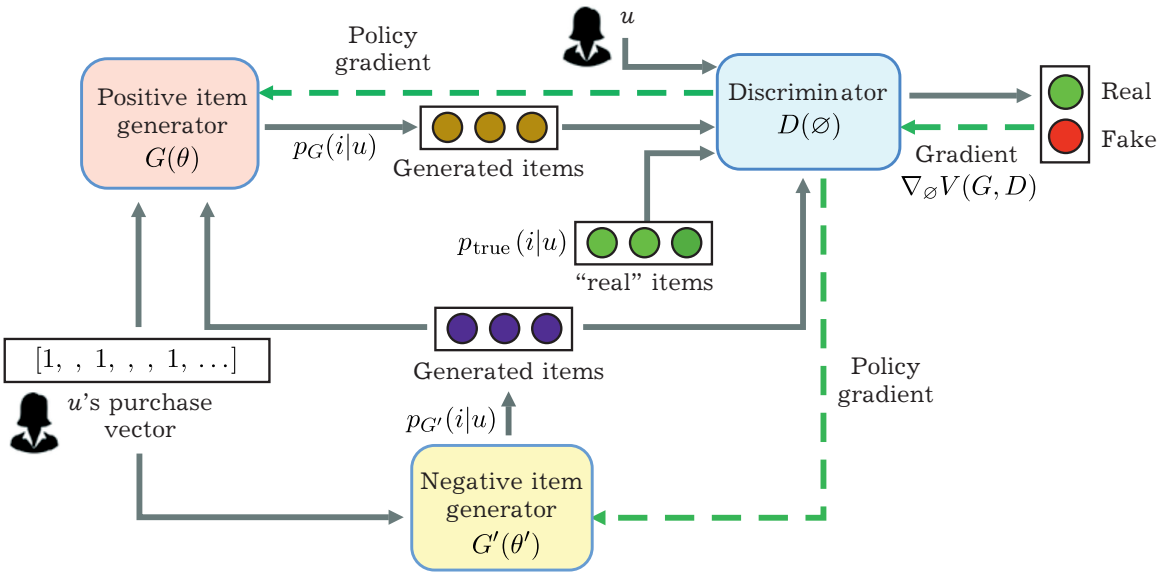

Fig. 4. Generator network casted as MF-based approach with embedding layers [15].

4. Similarity based models

4.1. Slim: sparse linear methods for top-N recommendation systems

SLIM (Sparse Linear Methods) [18,39], a model designed for solving the top-N recommendation problem. Despite being a model, SLIM operates more like a memorybased collaborative filtering (CF) technique. It builds a model of item-item similarity by optimizing certain criteria, including minimizing the Frobenius norm of the difference between a user-item interaction matrix and the product of item similarity matrix (SIMI) and a regularization term. The optimization process encourages sparsity using the 11-norm and ensures that similarity values are non-negative ($SIMI \geqslant 0$) while avoiding trivial solutions (diag(SIMI) = 0). Once SLIM learns the similarity matrix, it predicts user-item ratings by summing over items the product of a user's interaction history and the corresponding item similarity. This prediction method resembles item memorybased CF but differs in how the item-item similarity matrix is constructed. Finally, based on the predicted ratings, SLIM can recommend items that the user has not interacted with in the past, ordered by their predicted ratings in decreasing order, up to a specified limit N.

4.2. Item-based k-Nearest Neighbors (ItemKNN)

Item-based k-Nearest Neighbors [41] is a widely used model-based collaborative filtering technique in recommendation systems. It operates by creating connections between items based on user-item interactions. The core component of ItemKNN is the item-item similarity matrix, which measures the closeness or similarity between items. Common similarity metrics include cosine similarity, Pearson correlation, or Jaccard similarity, chosen based on dataset characteristics. The neighborhood size, denoted as 'k', is crucial in ItemKNN as it determines the number of similar items considered when making recommendations for an item. Smaller 'k' values lead to more localized recommendations, while larger 'k' values provide broader recommendations [14, 23]. The optimal 'k' depends on factors like data sparsity and desired personalization. One challenge in recommendation systems is popularity bias, where popular items dominate recommendations due to higher user interactions. ItemKNN addresses this by using a shrink term for regularization, reducing the impact of popularity, and ensuring less popular items get fair recommendations. Tuning the shrink term is vital for optimizing ItemKNN's performance. Fine-tuning ItemKNN involves systematically exploring various parameter configurations, including similarity metrics, neighborhood sizes (k), and shrink terms. This experimentation helps identify the best parameter setup for effective recommendations tailored to the dataset and context.

4.3. $P^3\alpha$ recommendation approach

The $\boldsymbol{P}^3\alpha$ recommendation approach is a sophisticated method designed for personalized recommendations. It operates within a graph-based framework, specifically utilizing a bipartite graph representing users and items. Users and items are nodes, and their interactions are edges in this graph. This graph-centric approach provides a structured representation of user-item relationships. $\boldsymbol{P}^3\alpha$ calculates item similarity through a unique method involving 2-step random walks within the bipartite graph, starting from user nodes [20]. This exploration helps quantify intricate item associations, crucial for generating relevant recommendations. To optimize performance, $\boldsymbol{P^3}\alpha$ fine-tunes two key parameters: neighborhood size (k), determining the scope of item consideration for recommendations, and similarity scaling coefficient (α), adjusting similarity values between items. These parameters play pivotal roles in tailoring recommendations to users' needs and preferences, making $\boldsymbol{P}^3\alpha$ a robust recommendation approach within a graph-based framework.

5. Matrix Completion based models

One would like to guess the missing data in a matrix or tensor. This problem is known as the matrix completion or tensor completion problem [42]. It comes up in a great number of applications including those of recommendation systems. Build such a system is the task of automatic predicting of the entries

in an unknown data matrix or tensor. A popular example is the movie recommendation case where the task is to make automatic predictions about the interests of a user by collecting taste information from his formal interests or by collecting them from other users. The direct representation is to model this problem as a matrix (for matrix completion). Users, rows of the matrix, are given the opportunity to rate items, columns of the data matrix. However, they usually rate very few ones so there are very few scattered observed entries of this data matrix. In this case, the users-ratings matrix is approximately low-rank because it is commonly believed that only very few factors contribute to an individual's tastes or preferences.

Authors in [43] proposed an approach for recommender systems where they used two step process. For each missing data given by a user u to an item i, the process runs a bi-clustering step to detect the users respectively items that share the most characteristics with u respectively i. It then constructs a sub-matrix containing the ratings that users who belong to u's cluster gave to items that belong to the i's cluster. The missing ratings are then obtained by using the Singular Value Thresholding algorithm SVT presented by [17]. Authors in [16], proposed an advanced version of this process by including a quantum implementation for the clustering process. This solution has a better time complexity.

The SVT algorithm objective is to minimize the rank of users-ratings. The rank minimization is a NP hard problem. However, it is possible to resolve it by convex programming. The rank function counts the number of non-vanishing singular values when the nuclear norm sums their amplitude. The nuclear norm is a convex function. It can be optimized efficiently via semi-definite programming.

Matrix completion-based approaches then minimizes the following problem:

- minimize $\|X\|_*$,
- subject to $P_\Omega(X) = P_\Omega(M)$,

where the nuclear norm $\|X\|_*$ is defined as the sum of its singular values:

$$\|X\|_* := \sum_i \sigma_i(X).$$

6. Discussion and experimental results

6.1. Datasets

In this study, we evaluate all the mentioned approaches using two well-established datasets within the RS community: MovieLens 1M, and MovieLens HetRec. The dataset statistics are detailed in Table 1. For this research, we had focused on implicit feedback only, by excluding movie ratings from the MovieLens datasets. As a result, we keep only the interactions between users and items, which collectively constitute the UserItem Rating Matrix (URM). This matrix is referred to as the full URM. To ensure a balanced distribution of data for both model training and evaluation, we have randomly divided each dataset into training and test sets, maintaining a 4:1 ratio, and we have considered only those users who have interacted with a minimum of 2 items. The test set is reserved exclusively for the final evaluation of all the algorithms (see Table 1).

Table 1. Dataset statistics.

Dataset	Interactions	Users	Items	Sparsity
ML 1M	1 000 209	6 040	3 706	95 53%
ML HetRec	855 598	2 113	1 0109	96 00%

6.2. Metrics

We use four metrics to compare results MAP, NDCG, PRECISION and RECALL.

— Mean Average Precision (MAP): MAP is a metric used in recommendation systems to measure how well the system ranks and suggests relevant items to users. It looks at the precision (the proportion of relevant items) in the top recommendations and calculates an average across multiple users or queries. A higher MAP indicates better recommendation quality.
Precision: Precision@ $k = \frac{(\text{NumberofRelevantItemsinTop-kRecommendations})}{k}$.
Average Precision (AP): $AP = \frac{1}{N}\sum_{k=1}^{N}$ Precision@k $\times$ Relevance@k.
Average Precision (MAP): $MAP = \frac{1}{Q}\sum_{i=1}^{Q} AP_i$.

— Normalized Discounted Cumulative Gain (NDCG): NDCG is another metric for recommendation systems that assesses the quality of recommendation lists. It considers both the relevance of items and their positions in the list. NDCG ranges from 0 to 1, with 1 indicating a perfect ranking where all relevant items are at the top. It is a measure of how well a recommendation system ranks and orders items for users.
Discounted Cumulative Gain (DCG): $DCG@k = \sum_{i=1}^{k} \frac{2^{rel_i-1}}{\log_2(i+1)}$, where: rel_i is relevance at position i.
Ideal Discounted Cumulative Gain (IDCG): $IDCG@k = \sum_{i=1}^{k} \frac{2^{rel(i)-1}}{\log_2(i+1)}$, where $rel_{(i)}$ is relevance when sorted.
Normalized Discounted Cumulative Gain (NDCG): $NDCG@k = \frac{DCG@k}{IDCG@k}$. NDCG ranges from 0 to 1, with 1 indicating a perfect ranking where all relevant items are top ranked.
— Recall is another important metric in the context of RS as it measures the ability of a model to capture all the relevant items. It is calculated as the ratio of the number of relevant items retrieved to the total number of relevant items. The formula for Recall@k, which considers the top-k recommendations, is as follows: Recall@k $k = \frac{(\text{NumberofRelevantItemsinTop-kRecommendations})}{TotalNumberofRelevantItems}$.

6.3. Results

Below, a comprehensive overview of the comparative analysis of all the approaches discussed in this research. The best results per dataset and per metric are given in bold, second-best results are underlined (see Tables 2 and 3). In our experiments, we have used hyper-parameters that align with state-of-the-art literature, as documented in [15–18, 20–22, 39–41, 43].

Table 2. Comparison of MAP and NDCG results over MovieLens 1M, and MovieLens HetRec.

Datasets	1M ML				HetRec			
Cutoff	@5		@20		@ 5		@20	
Models/Metrics	MAP	NDCG	MAP	NDCG	MAP	NDCG	MAP	NDCG
SLIM	0.3249	0.4298	0.2147	0.3775	0.471	0.5643	0.3284	0.4862
ItemKNN (cosine sim)	0.3121	0.4088	0.2063	0.3577	0.4783	0.5599	0.3215	0.4664
$\boldsymbol{P^3\alpha}$	0.3086	0.4066	0.2016	0.3553	0.4539	0.5432	0.3028	0.4532
PureSVD	0.3243	0.4197	0.2139	0.3644	0.5126	0.5933	0.3604	0.5020
WRMF	0.3200	0.4229	0.2178	0.3783	0.4859	0.5762	0.3393	0.4923
CAAE	0.1531	0.2217	0.0913	0.1941	0.3894	0.4767	0.2467	0.3902
CFGAN-user	0.3066	0.4044	0.1977	0.3487	0.4151	0.5123	0.2695	0.4224
GANMF-user	0.3551	**0.4564**	0.2423	**0.4032**	0.5255	0.6076	0.3715	**0.5151**
MC-CF	0.3867	0.4476	0.4298	0.2254	0.5391	0.6105	0.4298	0.3211
QMC-CF	**0.4105**	0.4509	**0.4306**	0.2065	**0.5547**	**0.6309**	**0.4306**	0.3084
SVT	0.2894	0.4106	0.3107	0.1907	0.4973	0.5783	0.4231	0.3587

Table 3. Comparison of PRECISION (Pres) and RECALL (Rec) results over MovieLens 1M, and MovieLens HetRec.

Datasets	1M ML				HetRec			
Cutoff	@5		@20		@5		@20	
Models/Metrics	PRES	RECA	PRES	RECA	PRES	RECA	PRES	RECA
SLIM	0.4051	0.1037	0.2817	0.2541	0.5405	0.0624	0.4279	0.1682
ItemKNN (cosine sim)	0.3880	0.0912	0.2743	0.2286	0.5414	0.0502	0.4155	0.1421
$\boldsymbol{P^3\alpha}$	0.3809	0.0904	0.2698	0.2289	0.5179	0.0536	0.3996	0.1434
PureSVD	0.3968	0.0915	0.2811	0.2279	0.5770	0.0563	0.4510	0.1541
CAAE	0.2091	0.0400	0.1552	0.1137	0.4574	0.0423	0.3453	0.1164
CFGAN-user	0.3817	0.0897	0.2654	0.2206	0.4933	0.0476	0.3736	0.1284
GANMF-user	0.4311	0.1070	0.3066	0.2629	0.5870	0.0582	0.4616	0.1592
MC-CF	**0.6398**	**0.4298**	**0.5011**	**0.3209**	**0.7452**	**0.498**	**0.6977**	**0.4010**
QMC-CF	0.6098	0.4193	0.481	0.3087	0.7259	0.4892	0.6741	0.3865
SVT	0.4161	0.2204	0.3924	0.2916	0.4996	0.3032	0.4812	0.3965

6.4. Discussion

The results in Tables 2 and 3 provide a comprehensive overview of the performance of various recommendation algorithms, based on MAP, NDCG, Precision, and Recall across two datasets MovieLens 1M and Hetrec2011, and cutoff values (@5 and @20). Analyzing the results, we observe that several recommendation models exhibit different levels of performance.

In terms of MAP and NDCG metrics, PureSVD, WRMF, $P^3\alpha$, ItemKNN (cosine similarity) and SLIM consistently provide commendable results. Specifically, these models achieve MAP scores ranging from 30.86% to 32.49% and NDCG scores ranging from 40.66% to 42.98% at the top 5 positions. However, their performance drops slightly when looking at the top 20, with MAP scores ranging from 20.16% to 21.78% and NDCG scores ranging from 35.53% to 37.38%. In contrast, CAAE performs lower, with MAP and NDCG in the top 5 and top 20. It should be noted that these models might require additional adjustments or might not be suitable for this dataset. Notably, GANMF-u, MC-CF and QMC-CF stand out as the top-performing models, systematically excelling in MAP and NDCG, especially in the top 5 positions, where their scores reach up to 45.64% for NDCG. However, SVT's performance fluctuates across different evaluation metrics, excelling in NDCG @5 but underperforming in NDCG @20.

Focusing on the HetRec dataset, SLIM, ItemKNN, $P^3\alpha$, PureSVD and WRMF demonstrate robust performance than on dataset 1M ML. This time, they get higher overall scores, indicating improved recommendations, with MAP and NDCG scores ranging from 45.39% to 56.43% for the top 5 positions. CAAE, although still lagging the top models, shows better results compared to its performance on the 1M ML dataset. GANMF-u appears to perform well, meaning its ability to provide high-quality recommendations on this dataset. However, MC-CF and QMC-CF also show strong performance, especially in the top 5 positions. Meanwhile, SVT's performance remains inconsistent, resembling trends seen in quality analysis.

Moving to Precision and Recall metrics, MC-CF and QMC-CF stand out as top-performing models, particularly excelling at higher cutoff values. These models consistently achieve high Precision and Recall scores, indicating their effectiveness in recommending relevant items to users. On the contrary, CAAE, CFGAN-user, and SVT show comparatively lower precision and recall values, suggesting need for improvement.

These results highlight the importance of selecting the right recommendation model and cutoff for specific datasets and the potential for further optimization in certain cases.

7. Conclusion

In conclusion, this survey reveals the richness of recommendation system techniques, from classical matrix factorization to cutting-edge GAN-based approaches. By tuning various parameters and understanding the nuances of each method, we can harness their power to provide tailored, high-quality recommendations. The diverse array of models discussed here underscores the need for a nuanced approach in choosing the right recommendation technique, considering factors such as data characteristics, application requirements, and computational resources. As the field of recommendation systems continues to evolve, this survey serves as a valuable resource for both researchers and practitioners seeking to navigate this dynamic landscape.

[1] Momoh F. O., Rakshit S., Vajjhala N. R. Exploratory study of machine learning algorithms in recommender systems. Proceedings of International Conference on Advanced Computing Applications. 571–580 (2021).

[2] Sharma R., Singh R. Evolution of recommender systems from ancient times to modern era: a survey. Indian Journal of Science and Technology. **9** (20), 1–12 (2016).

[3] Li S. S., Karahanna E. Online recommendation systems in a B2C E-commerce context: a review and future directions. Journal of the Association for Information Systems. **16** (2), 72–107 (2015).

[4] Beel J., Langer S., Nürnberger A., Genzmehr M. The Impact of Demographics (Age and Gender) and Other User-Characteristics on Evaluating Recommender Systems. Research and Advanced Technology for Digital Libraries. 396–400 (2013).

[5] Bhuvaneshwari P., Rao A. N., Robinson Y. H. Top-N Recommendation System Using Explicit Feedback and Outer Product Based Residual CNN. Wireless Personal Communications. **128**, 967–983 (2023).

[6] Filali-Zegzouti S., Banouar O., Benslimane M. Classification of Recommender systems using Deep Learning based generative models. Proceedings of the Statistics and Data Science Conference. 164–169 (2023).

[7] Van Meteren R., Van Someren M. Using content-based filtering for recommendation. Proceedings of the machine learning in the new information age: MLnet/ECML2000 workshop. **30**, 47–56 (2000).

[8] Afoudi Y., Lazaar M., Al Achhab M. Hybrid recommendation system combined content-based filtering and collaborative prediction using artificial neural network. Simulation Modelling Practice and Theory. **113**, 102375 (2021).

[9] Papadakis H., Papagrigoriou A., Panagiotakis C., Kosmas E., Fragopoulou P. Collaborative filtering recommender systems taxonomy. Knowledge and Information Systems. **64**, 35–74 (2022).

[10] Sarik G., Nematbakhsh M. A. Enhancing memory-based collaborative filtering for group recommender systems. Expert Systems with Applications. **42** (7), 3801–3812 (2015).

[11] Aditya P. H., Budi I., Munajat Q. A comparative analysis of memory-based and model-based collaborative filtering on the implementation of recommender system for E-commerce in Indonesia: A case study PT X. 2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS). 303–308 (2016).

[12] Movafegh Z., Rezapour A. Improving collaborative recommender system using hybrid clustering and optimized singular value decomposition. Engineering Applications of Artificial Intelligence. **126** (D), 107109 (2023).

[13] Ma Y., Liu Q. Generalized matrix factorization based on weighted hypergraph learning for microbe-drug association prediction. Computers in Biology and Medicine. **145**, 105503 (2022).

[14] Tran T., Lee K., Liao Y., Lee D. Regularizing matrix factorization with user and item embeddings for recommendation. Proceedings of the 27th ACM international conference on information and knowledge management. 687–696 (2018).

[15] Dervishaj E., Cremonesi P. GAN-based matrix factorization for recommender systems. Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing. 1373–1381 (2022).

[16] Ouedghiri O., Banouar O., Raghay S., El Haddaj S. Intelligent recommender system based on quantum clustering and matrix completion. Concurrency Computation Practice and Experience. **34** (15), e6943 (2022).

[17] Cai J.-E., Candès E. J., Zuowei S. A singular value thresholding algorithm for matrix completion. SIAM Journal on Optimization. **20** (4), 1956–1982 (2010).

[18] Christakopoulou E. Improving the Quality of Top-N Recommendation. University of Minnesota Ph.D. dissertation (2018).

[19] Anamisa D. R., Jauhari A., Mufarroha F. A. K-Nearest Neighbors Method for Recommendation System in Bangkalan's Tourism. ComTech: Computer, Mathematics and Engineering Applications. **14** (1), 33–44 (2023).

[20] Cooper C., Lee S. H., Radzik T., Siantos Y. Random walks in recommender systems: exact computation and simulations. Proceedings of the 23rd International Conference on World Wide Web. 811–816 (2014).

[21] Chae D.-K., Kang J.-S., Kim S.-W., Lee J.-T. CFGAN: A generic collaborative filtering framework based on generative adversarial networks. Proceedings of the 27th ACM International Conference on Information and Knowledge Management. 137–146 (2018).

[22] Chae D.-K., Shin J. A., Kim S.-W. Collaborative adversarial autoencoders: An effective collaborative filtering model under the GAN framework. IEEE Access. **7**, 37650–37663 (2019).

[23] Ghazarian S., Nematbakhsh M. A. Enhancing memory-based collaborative filtering for group recommender systems. Expert Systems with Applications. **42** (7), 3801–3812 (2014).

[24] Wang Q., Ji Z., Liu H., Zhao B. Deep Bayesian Multi-Target Learning for Recommender Systems. Preprint arXiv:1902.09154v1 (2019).

[25] Saberi D. M., Spina S. R. Deep Bayesian Recommendation Systems (2018).

[26] Rahman S. Extended Collaborative Filtering Recommendation System with Adaptive KNN and SVD. International Journal of Engineering and Management Research. **13** (4), 105–112 (2023).

[27] Kumar R., Verma B. K., Rastogi S. S. Social popularity based SVD++ recommender system. International Journal of Computer Applications. **87** (14), 33–37 (2014).

[28] Huang H. Enhancing Recommender Systems with Causal Inference Methodologies. UWSpace (2023).

[29] Lee D. D., Seung H. S. Learning the parts of objects by non-negative matrix factorization. Nature. **401** (6755), 788–791 (1999).

[30] Adyatma H. A., Baizal Z. K. A. Book Recommender System Using Matrix Factorization with Alternating Least Square Method. Journal of Information System Research. **4** (4), 1286–1292 (2023).

[31] Jannani A., Sael N., Benabbou F. Machine learning for the analysis of quality of life using the World Happiness Index and Human Development Indicators. Mathematical Modeling and Computing. **10** (2), 534–546 (2023).

[32] Tian B., Gu Y., Liu S. A Probability Matrix Factorization for User Behavior Perception Recommendation Model. 2023 IEEE 3rd International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA). 143–146 (2023).

[33] Zhang Z., Wu Q., Zhang Y., Liu L. Movie recommendation model based on probabilistic matrix decomposition using hybrid AdaBoost integration. PeerJ Computer Science. **9**, e1338 (2023).

[34] Liu N., Zhao J. Recommendation System Based on Deep Sentiment Analysis and Matrix Factorization. IEEE Access. **11**, 16994–17001 (2023).

[35] Kuchma M. I., Gatalevych A. I. Triangular form of Laurent polynomial matrices and their factorization. Mathematical Modeling and Computing. **9** (1), 119–129 (2022).

[36] Goodfellow I. J., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y. Generative adversarial nets. Proceedings of the 27th International Conference on Neural Information Processing Systems. **2**, 2672–2680 (2014).

[37] Qin W., Wu H., Lai Q., Wang C. A Parallelized, Momentum-incorporated Stochastic Gradient Descent Scheme for Latent Factor Analysis on High-dimensional and Sparse Ma-trices from Recommender Systems. 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC). 1744–1749 (2019).

[38] Creswell A., White T., Dumoulin V., Arulkumaran K., Sengupta B., Bharath A. A. Generative adversarial networks: An overview. IEEE Signal Processing Magazine. **35** (1), 53–65 (2018).

[39] Ning X., Karypis G. SLIM: Sparse Linear Methods for Top-N Recommender Systems. 2011 IEEE 11th International Conference on Data Mining. 497–506 (2011).

[40] Wang H., Wang J., Wang J., Zhao M., Zhang W., Zhang F., Li W., Xie X., Guo M. Learning graph representation with generative adversarial nets. IEEE Transactions on Knowledge and Data Engineering. **33** (8), 3090–3103 (2019).

[41] Deshpande M., Karypis G. Item-based top-N recommendation algorithms. ACM Transactions on Information Systems (TOIS). **22** (1), 143–177 (2004).

[42] Mohaoui S., El Qate K., Hakim A., Raghay S. Low-rank tensor completion using nonconvex total variation. Mathematical Modeling and Computing. **9** (2), 365–374 (2022).

[43] Banouar O., Raghay S. Enriching SPARQL Queries by User Preferences for Results Adaptation. International Journal of Software Engineering and Knowledge Engineering. **28** (08), 1195–1221 (2018).

Методи рекомендаційних систем на основі генеративних моделей і матричної факторизації: огляд

Філалі Зегзуті С.[1], Бануар О.[2], Бенсліман М.[1]

[1]*Лабораторія наук, техніки та управління, Університет Сіді Мохамеда Бен Абделлаха, Фес, Марокко*
[2]*Лабораторія інженерії та систем комп'ютерних наук, Університет Каді Айяда, Марракеш, Марокко*

Спільна фільтрація (CF) — це техніка, яка може відфільтрувати елементи, які можуть сподобатися користувачеві, на основі поведінки та вподобань подібних користувачів. Це ключовий метод, що сприяє створенню ефективної системи рекомендацій (RS). Системи рекомендацій на основі моделі, підмножина CF, використовують дані, зазвичай рейтинги, для побудови моделей для надання персоналізованих пропозицій користувачам. Мета цієї роботи полягає в тому, щоб надати всебічний огляд різних методів, що використовуються в модельному RS, зосереджуючись на їх теоретичних основах і практичному застосуванні. Досліджуються основні проблеми, пов'язані з рекомендаціями, включно з проблемою top-N рекомендацій, і досліджуються найсучасніші методи на основі моделей, які використовуються для вирішення цих проблем. У цьому огляді роздіено ці методи на три окремі класи: матрична факторізація, методи на основі подібності та методи на основі заповнення. Щоб порівняти їх ефективність, оцінено ці методи за наборами даних MovieLens за допомогою двох показників: середньої середньої точності (MAP), нормалізованого дисконтованого сукупного виграшу (NDCG), точності та повноти.

Ключові слова: *рекомендаційні системи; на основі моделі; top-N рекомендація; генеративно-змагальна мережа; матрична факторизація; заповнення матриці.*