# A review of mathematical approaches to recommendation algorithms: from collaborative filtering to deep learning

[1]Gjurgica Anastasov, [2]Mirjana Kocaleva Vitanova, [3]Biljana Zlatanovska, [4]Marija Miteva

[1,2,3,4] Faculty of Computer science
[1,2,3,4] Goce Delcev University, Stip, North Macedonia
[1]gurgica.210232@student.ugd.edu.mk, [2]mirjana.kocaleva@ugd.edu.mk,
[3]biljana.zlatanovska@ugd.edu.mk, [4]marija.miteva@ugd.edu.mk

***Abstract*—Recommendation algorithms are one of the most important technologies in modern information systems and are widely used in e-commerce, social networks, streaming systems and digital platforms. Their main goal is to provide personalized recommendations by analysing user preferences and interactions. These systems are based on mathematical concepts from linear algebra, optimization theory, statistics and machine learning. This paper presents an overview of the most important mathematical models and recommendation algorithms, with a special emphasis on collaborative filtration, matrix factorization and modern approaches based on deep learning. Their theoretical foundations, advantages, limitations and evaluation criteria are analysed. In addition, challenges related to data sparsity, the cold start problem and the computational complexity of the algorithms are considered. The paper provides a systematic overview of the development of recommender systems and identifies future research directions in this area.**

***Index Terms*— Recommendation algorithms, collaborative filtration, matrix factorization, machine learning, deep learning, recommender systems.**

## I. INTRODUCTION

The exponential growth of digital data has led to the emergence of a significant problem of information overload. Users of modern online platforms are faced with a huge number of available products, services and content, which makes it difficult to identify the most relevant information. In response to this challenge, recommendation algorithms have developed as a key technology that enables personalized filtering and ranking of content according to users' interests. The first research in the field of recommender systems began in the 1990s, and today they are an integral part of platforms such as Netflix, Amazon, YouTube and Spotify. Their goal is to predict which products, films, books or services might be of interest to a specific user based on his previous behaviour and the behaviour of other users. From a mathematical perspective, recommendation algorithms represent a complex modelling and optimization problem. Typically, interactions between users and objects are represented through a rating matrix, which in real conditions is extremely rare. Therefore, the central task is the reconstruction of missing values and the identification of latent factors that explain user preferences. In the last two decades, several approaches have been developed to solve this problem. Collaborative filtering is one of the most widespread methods and assumes that users with similar interests will show similar behaviour in the future. Later, matrix factorization allowed for a significant improvement in accuracy by revealing latent structures in the data. Modern research is increasingly focused on the application of deep neural networks and hybrid models that combine different sources of information. The aim of this paper is to provide a systematic overview of the mathematical foundations of recommendation algorithms, their development and contemporary trends. Special attention is paid to the mathematical modelling of interaction matrices, methods for measuring similarity, matrix factorization and the application of deep learning techniques. In addition, the most used evaluation metrics are analysed and the challenges faced by modern recommender systems are discussed.

## II. THEORETICAL FOUNDATIONS OF RECOMMENDATION ALGORITHMS

### *Conceptual foundations of recommendation algorithms*

Recommendation algorithms are intelligent mechanisms for filtering information whose main goal is to identify and suggest content, products or services that are likely to be relevant to a particular user. Their development is directly related to the problem of information overload, which arises from the exponential growth of digital data and content available through internet platforms [1]. According to Ricci and colleagues, recommender systems are software tools and techniques that provide personalized recommendations to users by analysing their interests, preferences and previous interactions [2]. Their role today is particularly important in e-commerce, streaming platforms, social networks, educational systems and digital marketing. In the most general case, a recommender system tries to estimate the function $f : U \times I \to R$ where: $U$ is the set of users, $I$ is the set of objects (films, books, products, etc.), $R$ is the set of possible ratings or preferences. The goal is to predict the value of the function for unknown combinations of users and items, i.e., to determine how likely a particular user is to show interest in a specific item [3]. In the literature, three main categories of recommender systems are most often distinguished [2], [4]:

- Content-Based Filtering,
- Collaborative Filtering,
- Hybrid Recommender Systems.

Content-based systems analyse the characteristics of items and recommend similar content to those that the user has previously used. Collaborative filtering is based on the behaviour of other users with similar interests, while hybrid models combine multiple techniques to improve accuracy and reduce the limitations of individual approaches [4].

***Mathematical model of interaction matrix***

Most modern recommendation algorithms are based on a matrix of interactions between users and items [5].

Let $R = \left[ r_{ui} \right]$ represent a rating matrix, where: $R \in \mathbb{R}^{m \times n}$ and:

- $m$ is the number of users,
- $n$ is the number of items,
- $r_{ui}$ is the rating that user u assigned to item $i$.

An example of such a matrix is:

| User | Movie 1 | Movie 2 | Movie 3 | Movie 4 |
|---|---|---|---|---|
| $U_1$ | 5 | ? | 3 | ? |
| $U_2$ | 4 | 2 | ? | 5 |
| $U_3$ | ? | 4 | 1 | ? |
| $U_4$ | 5 | ? | ? | 4 |

The symbol "?" indicates missing data. In real systems, more than 95% of the elements of the matrix are usually unknown [6]. This phenomenon is known as the sparsity problem and represents one of the biggest challenges in the development of recommender systems. Formally, the density of a matrix can be defined as:

$$Density = \frac{Number\ of\ known\ ratings}{m \times n}$$

while the sparsity is: $Sparsity = 1 - Density$. The higher the sparsity, the more difficult it becomes to predict missing ratings [7]. For this reason, most modern algorithms focus on discovering latent structures that allow the reconstruction of the matrix by minimizing an error function [8].

***Collaborative Filtration and the Statistical Basis of Similarity***

Collaborative filtration is the most widely used approach in recommender systems [9]. The basic assumption is that users who have had similar preferences in the past will exhibit similar behaviour in the future. There are two basic variants [10]: User-Based Collaborative Filtering and Item-Based Collaborative Filtering the User-Based approach calculates the similarity between users, while the Item-Based approach analyses the similarity between items. One of the most used measures is the cosine similarity:

$$\cos(x, y) = \frac{x \cdot y}{\| x \| \cdot \| y \|} \quad (1)$$

where: $x$ and $y$ are vectors of scores, $x \cdot y$ is the scalar product, and $\| x \|$ and $\| y \|$ are Euclidean norms. The values of the cosine similarity range in the interval:

$$-1 \le \cos(x, y) \le 1$$

with a value close to 1, users have a high level of similarity. In addition to cosine similarity, the Pearson correlation coefficient [11] is also often used in literature:

$$\cos(u, v) = \frac{\sum_i (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_i (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_i (r_{vi} - \bar{r}_v)^2}}$$

Where:

- $\bar{r}_u$ is the average rating of user $u$,
- $\bar{r}_v$ is the average rating of user $v$.

This metric considers the individual tendency of users to give higher or lower ratings. The prediction of the grade for a new subject is most often calculated through a weighted mean: $\bar{r}_{ui} = \frac{\sum_{v \in N(u)} \cos(u, v) r_{vi}}{\sum_{v \in N(u)} | \cos(u, v) |}$ where $N(u)$ represents the set of most similar users. Despite their popularity, collaborative models face several limitations [12]: the cold start problem, high data sparsity, scalability in large systems, sensitivity to noise and inaccurate grades. For these reasons, modern systems increasingly use matrix factorization and models based on machine and deep learning, which will be discussed in the following chapters.

## III. Matrix Factorization and Mathematical Optimization

### *Basics of Matrix Factorization*

Matrix factorization is one of the most important techniques in modern recommendation algorithms. Its popularity has increased significantly after the Netflix Prize competition, where models based on matrix factorization achieved the best results [13]. The basic idea is that the complex interaction between users and items can be represented by a smaller number of latent factors that describe their hidden characteristics. Let the rating matrix be represented by $R \in \mathbb{R}^{m \times n}$ where m is the number of users, n is the number of items, $r_{ui}$ is the rating that user u assigned to item $i$. The goal is to approximate the matrix R as the product of two matrices of lower dimension:

$$R \approx PQ^T \quad (2)$$

where $P \in \mathbb{R}^{m \times k}$ is the matrix of latent factors of users and $Q \in \mathbb{R}^{n \times k}$ is the matrix of latent factors of objects. The parameter *k* represents the dimension of the latent space and is usually significantly smaller than the number of users and objects. Each row of the matrix $P$ describes the user profile through latent features, while each row of the matrix $Q$ represents the latent properties of the object [14].

The predicted score is calculated as: $\bar{r}_{ui} = p_u^T q_i$ where:

- $p_u$ is the latent vector of the user,
- $q_i$ is the latent vector of the object.

This model allows the discovery of hidden relationships between users and objects without explicitly defining their features.

### *Singular Value Decomposition (SVD)*

One of the most important techniques in linear algebra applied to recommender systems is singular value decomposition (SVD). According to the SVD theorem, any matrix $R$ can be represented as $R = U\Sigma V^T$ where:

- $U$ is an orthogonal matrix of left singular vectors,
- $V$ is an orthogonal matrix of right singular vectors,
- $\Sigma$ is a diagonal matrix containing the singular values [15].

The singular values are ordered by size: $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r$ where $r$ is the rank of the matrix. In recommender systems, a reduced SVD approximation is most often used: $R_k = U_k \Sigma_k V_k^T$ where only the most significant singular values are retained. In this way, noise is eliminated and the most important information contained in the data is retained [13]. The main advantages of SVD are dimensionality reduction, detection of latent structures, improvement of generalization. The disadvantage is that classical SVD requires a complete matrix and does not work directly for very sparse matrices, which is a typical case in recommender systems [16].

### *Modern Matrix Factorization Models*

#### *Basic Matrix Factorization*

The basic factorization model is based on minimizing the error between the actual and predicted scores: $r_{ui} \approx p_u^T q_i$. This model is simple and efficient but does not consider the individual characteristics of users and items.

#### *Probabilistic Matrix Factorization (PMF)*

Mnih and Salakhutdinov [17] proposed Probabilistic Matrix Factorization (PMF), in which the latent factors are modelled as normally distributed random variables.

The probability function is:

$$P(R \mid P, Q, \sigma^2) = \prod_{(u,i) \in K} N(r_{ui} \mid p_u^T q_i, \sigma^2)$$

where:

- $N$ is a normal distribution,
- $K$ is the set of known scores.

PMF allows for better handling of noise and missing data.

#### *SVD++*

Koren [18] extends the classical model by introducing implicit feedback. The prediction becomes:

$$\bar{r}_{ui} = \mu + b_u + b_i + q_i^T \left( p_u + \mid N(u) \mid^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right)$$

where:

- $\mu$ is the global average,
- $b_u$ and $b_i$ are user and item biases,

- $N(u)$ is the set of user interactions.

This model was among the most successful in the Netflix Prize competition [13].

***Non-negative Matrix Factorization (NMF)***

In NMF all elements are non-negative: $R \approx WH$ where: $W \geq 0,\ H \geq 0.$ This approach provides better interpretability because the factors have physical meaning [19].

***Loss Function and Regularization***

The goal of any model is to minimize the loss function. The most used function is: $L = \sum_{(u,i)\in K} (r_{ui} - p_u^T q_i)^2$. This function measures the squared error between the actual and predicted scores. To avoid overfitting, L2 regularization is added: $L = \sum_{(u,i)\in K} (r_{ui} - p_u^T q_i)^2 + \lambda\left(\| p_u \|^2 + \| q_i \|^2\right)$ where $\lambda$ is the regularization parameter. Regularization constrains the parameter values and improves the model's ability to generalize to new data [20].

***Optimization Methods***

***Gradient Descent***

Gradient descent is an iterative technique for minimizing the loss function [21]. The general formula is $\theta_{t+1} = \theta_t - \gamma \nabla L(\theta_t)$ where $\gamma$ is the learning rate.

***Stochastic Gradient Descent (SGD)***

The most used technique in recommender systems is stochastic gradient descent. The error is:

$$e_{ui} = r_{ui} - \bar{r}_{ui} = r_{ui} - p_u^T q_i \quad (3)$$

The parameter update is:

$$p_u \leftarrow p_u + \gamma (e_{ui}\, q_i - \lambda\, p_u) \quad (4)$$

$$q_i \leftarrow q_i + \gamma (e_{ui}\, p_u - \lambda\, q_i) \quad (5)$$

The process is repeated until the loss function reaches a minimum [13].

***Alternating Least Squares (ALS)***

ALS is an alternative optimization method that alternately optimizes the matrices $P$ and $Q$ [22]. The main advantages are stable convergence, good parallelization, efficiency on large data sets.

***Advantages and Limitations of Matrix Factorization***

Matrix factorization has become one of the most widely used techniques in recommender systems due to its ability to effectively model user–item interactions and generate accurate recommendations. One of its main advantages is its high predictive accuracy, which has been demonstrated in numerous practical applications and benchmark competitions such as the Netflix Prize [13]. Furthermore, matrix factorization handles sparse rating matrices efficiently, making it particularly suitable for real-world recommendation datasets where only a small fraction of possible ratings is available. Another important advantage is its ability to discover latent factors that capture hidden relationships between users and items. These latent representations enable the model to identify underlying preferences and characteristics without requiring explicit information about users or products. In addition, matrix factorization offers excellent scalability, allowing it to be applied to large-scale recommendation systems with millions of users and items [13]. Despite its strengths, matrix factorization also has several limitations. One of the most significant challenges is the cold-start problem, which occurs when new users or items have insufficient interaction data to estimate reliable latent factors. The effectiveness of the method also depends on the availability of a sufficient number of users–item interactions, as sparse data can reduce recommendation quality. Another limitation is the complexity of parameter selection. The performance of the model is highly dependent on parameters such as the number of latent factors, learning rate, and regularization coefficients, which often require extensive tuning. Additionally, the latent factors learned by the model are not always easily interpretable, making it difficult to explain the reasons behind specific recommendations [3]. Despite these limitations, matrix factorization remains one of the most successful and influential techniques for building recommender systems due to its strong predictive performance, scalability, and ability to uncover hidden patterns in user behaviour [30], [31].

***Example***

In this section, we will illustrate the mathematical background of recommendation algorithms through an example. Due to the scope of the paper, an application of cosine similarity in a recommendation system and matrix factorization using the SGD (Stochastic Gradient Descent) method will be given.

The system comprises five registered users

$$A = (3,2,0,1), B = (3,2,1,0), C = (5,4,3,0), D = (0,1,3,5), E = (2,5,1,0)$$

each expressing preferences across four movie genres: action, comedy, drama, and thriller. Each user is modelled as a vector in a multidimensional feature space, where each coordinate represents the level of preference for a particular genre. Specifically, the first

coordinate corresponds to the preference for action movies, the second to comedy movies, the third to drama movies, and the fourth to thriller movies.

The intensity of each user's preference for a particular movie genre is represented by numerical values based on a six-point Likert scale, as presented in Table 1.

Table 1. The intensity of interest via Likert scale

| Number values | Description |
|---|---|
| 0 | never watches |
| 1 | very rarely watches |
| 2 | rarely watches |
| 3 | occasionally watches |
| 4 | frequently watches |
| 5 | loves watching |

Based on this scale, the preferences of users A, B, C, D, and E with respect to the four movie genres are summarized in Table 2.

Table 2. User preferences according to movie genre

| User | Action | Comedy | Drama | Thriller |
|---|---|---|---|---|
| $A$ | 3-occasionally watches | 2-rarely watches | 0-never watches | 1-very rarely watches |
| $B$ | 3-occasionally watches | 2-rarely watches | 1-very rarely watches | 0-never watches |
| $C$ | 5-loves watching | 4-frequently watches | 3-occasionally watches | 0-never watches |
| $D$ | 0-never watches | 1-very rarely watches | 3-occasionally watches | 5-loves watching |
| $E$ | 2-rarely watches | 5-loves watching | 1-very rarely watches | 0- never watches |

**The cosine similarity in a recommendation system.** The similarity between users' preferences is measured using the cosine similarity metric, defined by Equation (1). The computed cosine similarity values for all pairs of users are presented in Table 3.

Table 3. The cosine similarity metric between users' preferences

| The couple of users | The cosine similarity |
|---|---|
| $(A,B)$ | 0,93 |
| $(A,C)$ | 0,87 |
| $(A,D)$ | 0.26 |
| $(A,E)$ | 0.61 |
| $(B,C)$ | 0.98 |
| $(B,D)$ | 0.19 |
| $(B,E)$ | 0.83 |
| $(C,D)$ | 0.31 |
| $(C,E)$ | 0.85 |
| $(D,E)$ | 0.25 |

As shown in Table 3, the highest similarity is observed between users B and C (0.98), followed by users $A$ and $B$ (0.93). High cosine similarity values are also obtained for the pairs $(A,C)$ (0.87), $(B,E)$ (0.83), and $(C,E)$ (0.85), indicating that these users exhibit comparable preference patterns across the considered movie genres. In contrast, user $D$ demonstrates substantially different preferences from the remaining users, as evidenced by the considerably lower cosine similarity values.

Based on these results, the recommender system can generate personalized recommendations using a collaborative filtering approach, whereby each user is recommended movie genres preferred by other users with similar preference profiles. Consequently, users $A$, $B$, $C$, and $E$ form a group characterized by relatively high mutual similarity, making them suitable candidates for recommendation sharing. Conversely, user $D$ constitutes a distinct preference profile and is therefore less likely to benefit from recommendations generated from the preferences of the other users.

**SGD (Stochastic Gradient Descent) method.** The initial preference matrix is

$$R=\begin{bmatrix}3&2&0&1\\3&2&1&0\\5&4&3&0\\0&1&3&5\\2&5&1&0\end{bmatrix}_{5\times4}.$$

For the matrix factorization process, two latent factors are assumed (i.e., $k=2$). The first latent factor represents users' preferences for action and comedy movies, whereas the second latent factor captures their preferences for drama and thriller movies. Although these latent factors are not directly observable, they describe hidden characteristics that explain the underlying structure of users' preferences.

In the SGD algorithm, the user-factor matrix $P$ and the item-factor matrix $Q$ are typically initialized with small random values drawn from the interval [0,1]. In this study, the learning rate and the regularization parameter were empirically set to $\gamma=0.01$ and $\lambda=0.02$, respectively. The learning rate controls the magnitude of parameter updates during optimization, whereas the regularization parameter prevents overfitting by penalizing excessively large latent factor values.

$$P=\begin{bmatrix}0.5&0.3\\0.6&0.2\\0.8&0.4\\0.2&0.9\\0.7&0.5\end{bmatrix}_{5\times2},\quad Q=\begin{bmatrix}0.9&0.2\\0.8&0.3\\0.3&0.8\\0.2&0.9\end{bmatrix}_{4\times2}.$$

Starting from the initial random matrices, the SGD algorithm iteratively updates the latent factors by minimizing the squared prediction error between the observed and the predicted preference values. During each iteration, the latent factors associated with both users and movie genres are adjusted according to the prediction error until the optimization process converges. Upon convergence, the resulting matrices $P$ and $Q$ provide latent representations of users and movie genres, respectively. These learned latent representations are subsequently used to estimate unknown preference values and generate personalized movie recommendations. In our example we will use them to generate personalized movie recommendations.

First epoch.

*First estimate*. The first estimate for the first user $A$ and for the action genre of the matrix $R$ is $r_{11}=3$. The user vector $A$ and the vector for the action genre are $p_A=(0.5,0.3)^T$ and $q_{Action}=(0.9,0.2)^T$ correspondingly. The predicted value is $\tilde{r}_{11}=0.51$ with the error $e_{11}=r_{11}-\tilde{r}_{11}=2.49$, (3). The first and second element of the updating matrix $P$ are $p_{11}=0.52231$ and $p_{12}=0.30492$, (4).The new vector for the first user $A$ is $p_A=(0.52231,0.30492)^T$. The first and second element of the updating matrix $Q$ are $q_{11}=0.91227$ and $q_{12}=0.20743$, (5). The new vector for the action genre is $q_{Action}=(0.91227,0.20743)^T$.

*Second estimate*. The next estimate for the first user $A$ and for the comedy genre of the matrix $R$ is $r_{12}=2$ and the same process is repeated.

After for the next estimate $r_{13}=0$, we are continuing for $r_{14}=1$, etc.

By using the programming language Python according to equation (2), we obtained the updating matrices $P$ and $Q$:

$$P = \begin{bmatrix} 0.53448 & 0.312 \\ 0.63472 & 0.21176 \\ 0.86984 & 0.43288 \\ 0.21624 & 0.95592 \\ 0.7456 & 0.51312 \end{bmatrix}, \quad Q = \begin{bmatrix} 0.96856 & 0.23218 \\ 0.87324 & 0.34693 \\ 0.32864 & 0.83168 \\ 0.20132 & 0.93343 \end{bmatrix},$$

$$R \approx PQ^T = \begin{bmatrix} 0.5901 & 0.575 & 0.4351 & 0.3988 \\ 0.6639 & 0.6277 & 0.3847 & 0.3254 \\ 0.943 & 0.9098 & 0.6459 & 0.5792 \\ 0.4314 & 0.5205 & 0.8661 & 0.9358 \\ 0.8413 & 0.8291 & 0.6718 & 0.6291 \end{bmatrix}.$$

After the first epoch, the updated matrices $P$ and $Q$ are obtained. The corresponding value of the loss function is $L = 88.972467$.

After 5000 iterations SGD converges to $R \approx PQ^T$. Then we would obtain something of the form:

$$P = \begin{bmatrix} 0.99716293 & 0.39048917 \\ 1.14556672 & 0.24310339 \\ 2.08126155 & 0.70060943 \\ -0.34853998 & 2.37368389 \\ 1.55153409 & 0.39444047 \end{bmatrix}, \quad Q = \begin{bmatrix} 2.1005877 & 0.33460072 \\ 2.03449945 & 0.69531344 \\ 0.57986632 & 1.34398118 \\ -0.50337543 & 2.03010679 \end{bmatrix},$$

$$R \approx PQ^T = \begin{bmatrix} 2.2253 & 2.3002 & 1.103 & 0.2908 \\ 2.4877 & 2.4997 & 0.991 & -0.0831 \\ 4.6063 & 4.7215 & 2.1485 & 0.3747 \\ 0.0621 & 0.9413 & 2.9881 & 4.9943 \\ 3.3911 & 3.4309 & 1.4298 & 0.0198 \end{bmatrix}.$$

This immediately shows:

- $A$ and $B$ are close;
- $C$ is close to $A$ and $B$;
- $E$ is moderately close to them;
- $D$ is very different.

Therefore, SGD would reach a conclusion like cosine analysis, but through latent factors rather than a direct comparison of scores.

The analysis shows that users $A, B, C$, and $E$ form a group with related interests, while user $D$ represents a distinct profile. Such results allow for the generation of more accurate recommendations compared to approaches that rely solely on direct cosine similarity.

## IV. Deep Learning and Modern Approaches to Recommendation Algorithms

### *Neural Collaborative Filtering (NCF)*

Traditional collaborative filtering models are mostly based on linear operations as a scalar product between latent vectors of users and items. However, real user preferences often present nonlinear relationships that cannot be fully captured by classical matrix factorization models. In response to this limitation, He et al. [23] propose the Neural Collaborative Filtering (NCF) model, which uses multilayer neural networks to model the interactions between users and items. The basic idea is to feed the latent vectors of the user and items as input to a neural network:

$$\{\hat{y}\}_{\{ui\}} = f(P_u, Q_i)$$

where the function (f) is approximated by a multilayer neural architecture. In this way, the model can learn complex nonlinear dependencies that are not available in classical matrix factorization.

### *Autoencoders in recommender systems*

An autoencoder is a specialized neural network whose goal is to reconstruct input data through a compressed latent representation [24]. The architecture consists of: Encoder, Bottleneck Layer, Decoder. Mathematically: $z = f(x), \{\hat{x}\} = g(z),\ x = g(z)$ where:

- $(x)$ is the original input,
- $(z)$ is the latent representation,
- $\{\hat{x}\}$ is the reconstructed output.

In recommender systems, the input vector represents the user's ratings, while the reconstructed output represents the prediction of the missing ratings. Autoencoder models show high efficiency when working with sparse data and large interaction matrices.

***DeepFM***

DeepFM is a model proposed by Guo et al. [25] that combines Factorization Machines and Deep Neural Networks. The output of the model is: $y = FM(x) + DNN(x)$ where:

- The FM component models are low-order interactions,
- The DNN component detects complex high-order nonlinear interactions.

DeepFM is particularly popular in e-commerce and advertising recommendation systems because it allows for the simultaneous modelling of explicit and implicit features.

***Transformer-Based Recommender Systems***

Following the success of the Transformer architecture in natural language processing, it has also been applied to recommender systems [26]. This allows the model to identify the most relevant previous interactions when generating a recommendation. Transformer models enable better modelling of sequential behaviour; better understanding of temporal dependencies and better performance on large datasets.

***Graph Neural Networks (GNN)***

Many modern systems represent users and objects as nodes in a graph. Let the graph be defined as: $G = (V, E)$ where: $(V)$ is the set of nodes and $(E)$ is the set of links. Graph Neural Networks enable information transfer through the graph through iterative updating:

$$h_v^{(k+1)} = \sigma\left( W \sum_{u \in N(v)} h_u^{(k)} \right)$$

where:

- ( $h_u$ ) is the representation of the node,
- ( $N(v)$ ) is the set of neighbouring nodes.

GNN models show exceptionally good results in systems with complex interactions between users and objects [27], [28], [29].

## V. METRICS FOR EVALUATING RECOMMENDATION ALGORITHMS

Evaluating recommender systems is an important part of their development. The most used metrics are MAE, RMSE, Precision, Recall, and NDCG.

***Mean Absolute Error (MAE)***

Mean Absolute Error (MAE) measures the average absolute prediction error between the actual values and the predicted values:

$$MAE = \frac{1}{N} \sum_{i=1}^{N} | r_i - \overline{r_i} |$$

where $r_i$ represents the actual rating, $\overline{r_i}$ represents the predicted rating, and $N$ denotes the total number of observations. A lower MAE value indicates higher prediction accuracy and better model performance.

***Root Mean Square Error (RMSE)***

RMSE gives more weight to larger errors:

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (r_i - \overline{r_i})^2}$$

Usually, RMSE is a more stringent metric than MAE.

***Precision***

Precision measures the accuracy of the recommended items by calculating the proportion of relevant recommendations among all recommended items:

$$\text{Precision} = \frac{\{TP\}}{\{TP + FP\}}$$

where:

- **TP (True Positives)** represents the number of relevant items that were correctly recommended.
- **FP (False Positives)** represents the number of non-relevant items that were incorrectly recommended.

A higher Precision value indicates that the recommender system generates more relevant recommendations and fewer irrelevant ones.

***Recall***

Recall measures the system's ability to find all relevant items:

$$\text{Recall} = \frac{\{TP\}}{\{TP + FN\}}$$

where FN represents missed relevant items.

***Normalized Discounted Cumulative Gain (NDCG)***

NDCG evaluates the accuracy of the ranking:

$$DCG_p = \sum_{\{i=1\}}^{\{p\}} \frac{rel_i}{\log_2(i+1)}$$

$$\text{NDCG} = \frac{DCG_p}{IDCG} \quad \text{NDCG} = \frac{DCG_p}{IDCG}$$

A value close to 1 indicates optimal ranking.

## VI. Discussion and Comparative Analysis

The development of recommendation algorithms shows a clear evolution from simple statistical models to complex architecture based on deep learning. Collaborative filtering is the basis of modern recommender systems due to its simplicity and intuitiveness. However, problems related to data sparsity and cold start limiting its application in large systems. Matrix factorization represents a significant step forward because it allows for the discovery of latent factors and better handling of sparse matrices. The success of the Netflix Prize has shown that these models can achieve high accuracy even with high-dimensional data. Modern models based on deep learning further overcome the limitations of linear models. Neural Collaborative Filtering, Autoencoder architectures, DeepFM models and Transformer-based systems allow the discovery of complex nonlinear patterns that cannot be modelled through classical techniques. Graph Neural Networks represent one of the most promising directions for future research because they allow for natural modelling of complex relationships between users and objects. However, their computational complexity remains a significant challenge. According to the analysed literature, there is no universally best algorithm. The choice of model depends on the nature of the data, available resources and the specific application. In practice, hybrid models that combine multiple approaches are increasingly used to achieve better accuracy and robustness.

## VII. Conclusion

Recommendation algorithms are one of the most important technologies in modern information systems. Their application in e-commerce, multimedia platforms, social networks and educational systems enables personalized user experience and more efficient use of information. The paper analysed the mathematical foundations of recommender systems, starting from collaborative filtration and matrix factorization, up to modern models based on deep learning. Special attention was paid to the sparsity problem, optimization techniques and evaluation methods. The results of the analysis show that matrix factorization remains one of the most successful recommendation techniques, while modern neural architecture allows for further improvement of accuracy by modelling complex nonlinear interactions. At the same time, Graph Neural Networks and Transformer architectures represent a current research direction with significant potential for further development. In the future, increased use of explainable recommender systems, graphical models, and generative artificial intelligence is expected, which will further expand the application and efficiency of recommendation algorithms.

## References


[1] G. Adomavicius and A. Tuzhilin, “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*, vol. 17, no. 6, pp. 734–749, 2005.

[2] F. Ricci, L. Rokach, and B. Shapira, *Recommender Systems Handbook*, 3rd ed. New York: Springer, 2022.

[3] C. C. Aggarwal, *Recommender Systems: The Textbook*. Cham: Springer, 2016.

[4] M. Pazzani and D. Billsus, “Content-based recommendation systems,” in *The Adaptive Web*, Springer, 2007, pp. 325–341.

[5] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[6] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, “Recommender systems survey,” *Knowledge-Based Systems*, vol. 46, pp. 109–132, 2013.

[7] X. Su and T. M. Khoshgoftaar, “A survey of collaborative filtering techniques,” *Advances in Artificial Intelligence*, vol. 2009, Article ID 421425, 2009.

[8] N. Halko, P. Martinsson, and J. Tropp, “Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions,” *SIAM Review*, vol. 53, no. 2, pp. 217–288, 2011.

[9] J. Schafer, J. Konstan, and J. Riedl, “Collaborative filtering recommender systems,” in *The Adaptive Web*, Springer, 2007, pp. 291–324.

[10] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-based collaborative filtering recommendation algorithms,” in *Proceedings of WWW 2001*, pp. 285–295.

[11] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl, “An algorithmic framework for performing collaborative filtering,” *Proceedings of SIGIR*, pp. 230–237, 1999.

[12] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender Systems: An Introduction*. Cambridge University Press, 2011.

[13] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” Computer, vol. 42, no. 8, pp. 30–37, 2009.

[14] C. C. Aggarwal, Recommender Systems: The Textbook. Springer, 2016.

[15] G. H. Golub and C. F. Van Loan, Matrix Computations, 4th ed. Baltimore: Johns Hopkins University Press, 2013.

[16] S. Funk, “Netflix Update: Try This at Home,” 2006.

[17] A. Mnih and R. Salakhutdinov, “Probabilistic matrix factorization,” in Advances in Neural Information Processing Systems (NIPS), 2008, pp. 1257–1264.

[18] Y. Koren, “Factorization meets the neighborhood: A multifaceted collaborative filtering model,” Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 426–434, 2008.

[19] D. D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” Advances in Neural Information Processing Systems, vol. 13, pp. 556–562, 2001.

[20] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning, 2nd ed. Springer, 2009.

[21] S. Boyd and L. Vandenberghe, Convex Optimization. Cambridge University Press, 2004.

[22] Y. Hu, Y. Koren, and C. Volinsky, “Collaborative filtering for implicit feedback datasets,” Proceedings of the IEEE International Conference on Data Mining, pp. 263–272, 2008.

[23] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, “Neural Collaborative Filtering,” Proceedings of WWW 2017, pp. 173–182, 2017.

[24] S. Sedhain, A. Menon, S. Sanner, and L. Xie, “AutoRec: Autoencoders Meet Collaborative Filtering,” Proceedings of WWW 2015, pp. 111–112, 2015.

[25] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, “DeepFM: A Factorization-Machine Based Neural Network for CTR Prediction,” IJCAI 2017, pp. 1725–1731, 2017.

[26] W.-C. Kang and J. McAuley, “Self-Attentive Sequential Recommendation,” IEEE International Conference on Data Mining (ICDM), pp. 197–206, 2018.

[27] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, “Neural Graph Collaborative Filtering,” Proceedings of SIGIR 2019, pp. 165–174, 2019.

[28] C. C. Aggarwal, Recommender Systems: The Textbook, Springer, 2016.

[29] F. Ricci, L. Rokach, and B. Shapira, Recommender Systems Handbook, 3rd ed., Springer, 2022.

[30] Karamazova Gelova E., Kocaleva M., Kertakova M. (2021). Statistical analysis of student achievement using different ways of learning. In: 4th TSD Conference, 18 Dec 2020, Skopje, Macedonia.

[31] Jordeva S., Anusheva H., Golomeova S., Zhezhova S., Dimitrijeva V., Mojsov K., Kertakova M. (2021). A cut marker for aircraft seat cover. Tekstilna industrija, 69 (2). pp. 40-47. ISSN 0040-2389