This study's object is the process that improves efficiency and accuracy in delivering personalized recommendations to users in systems based on reinforcement learning.

The principal task addressed in the study is to improve recommendation adaptation and personalization by assigning a dedicated agent to each user. This approach reduces the influence of other users' activity and allows for more precise modeling of individual preferences.

The proposed approach employs an Actor–Critic model implemented using the Deep Deterministic Policy Gradient algorithm to achieve more stable training and maximize long-term rewards in sequential decision-making processes. Recommendations are generated using the unique characteristics of items that are based on users' historical interactions. Neural networks are trained with separate parameter configurations for single-agent and multi-agent models.

Experimental results on the MovieLens dataset demonstrate the superiority of the multi-agent model over the single-agent baseline across key evaluation metrics. For top-5 recommendations, the multi-agent model achieved improvements of + 4% for Precision@5; + 0.32% for Recall@5; and + 2.92% in Normalized Discounted Cumulative Gain NDCG@5. For top-10 recommendations, gains were + 1% for Precision@10; + 0.18% for Recall@10; and + 1.14% for NDCG@10, respectively.

Simulations for individual users showed that the multi-agent model outperformed the single-agent baseline in 66 out of 100 cases in terms of cumulative reward. The proposed system demonstrates effectiveness in capturing user preferences, improving recommendation quality, and adapting to evolving user preferences over time.

The main area of practical application for the results includes dynamic online environments such as e-commerce systems, media platforms, social networks, and news aggregators

Keywords: personalized recommendation, reinforcement learning, multi-agent environment, Actor-Critic model

UDC 004.85:004.89

DOI: 10.15587/1729-4061.2025.340491

DEVELOPMENT OF A MULTI-AGENT ADAPTIVE RECOMMENDATION SYSTEM BASED ON REINFORCEMENT LEARNING

Bohdan Romaniuk
Corresponding author
PhD Student*
E-mail: bohdan.romaniuk@lnu.edu.ua
Olha Peliushkevych
PhD, Associate Professor*
*Department of Discrete Analysis and Intelligent System
Ivan Franko National University of Lviv
Universytetska str., 1, Lviv, Ukraine, 79000

Received 04.07.2025
Received in revised form 04.09.2024
Accepted date 15.09.2025
Published date 31.10.2025

How to Cite: *Romaniuk, B., Peliushkevych, O. (2025). Development of a multi-agent adaptive recommendation system based on reinforcement learning. Eastern-European Journal of Enterprise Technologies, 5 (2 (137)), 43–54. https://doi.org/10.15587/1729-4061.2025.340491*

1. Introduction

In today's digital world, the rapid growth of online services such as media platforms, news resources, and e-commerce has led to a significant increase in the amount of information and content available to users. Thousands of new books, films, games, and other types of digital products are released every year, making it much more difficult to select relevant content based on users' individual interests and preferences. This volume of information requires significant time and intellectual resources to analyze and compare, making the decision-making process laborious and inefficient. Whereas a decade ago, choosing a book, TV series, or electronic device was easy due to the limited number of available options, today consumers must carefully analyze reviews, comments, specifications, and other details to make the best decision. To overcome these difficulties, recommendation systems employing a wide range of algorithmic approaches are being actively developed and implemented. Such systems analyze the history of users' interactions with information resources and provide personalized recommendations. Recommendation systems play a key role in digital business, as they can influence consumer behavior and, consequently, increase company profits. Virtually all modern digital platforms use recommendation algorithms to increase profitability and improve user experience.

In this context, recommendation systems based on reinforcement learning and deep neural networks are particularly promising. These approaches are gradually replacing conventional approaches such as matrix factorization [1], content-based filtering [2], Naive Bayes classifier [3], and multi-armed bandit methods [4]. The main limitations of classical methods are the assumption of static user preferences, which are actually dynamic, and the focus on short-term interactions (e.g., clicks or likes), which often ignore the long-term value of user activity. Reinforcement learning allows one to formalize the recommendation process as a sequential decision-making problem, within which the system learns to optimize the reward based on long-term interactions with users. Reinforcement learning algorithms are classified into two categories: policy-based, which study the correspondence between states and actions, and value-based, which determine the expected reward for actions and choose the most beneficial one. In order to combine the advantages of both paradigms, the Actor-Critic model [5] was devised, which combines policy optimization with value estimation. Deep Deterministic Policy Gradient approaches (DDPG) [6] are applied to recommendation problems by representing the states of users and objects within a continuous vector space. These methods and approaches have gained considerable popularity due to their ability to analyze consistent user behavior and adapt to changes in their interests. Leading technology companies such as Google, Microsoft, Amazon, and Netflix are actively implementing these approaches, providing effective recommendations. Machine learning demonstrates exceptional capabilities in environments where dynamic modeling and planning of long-term engagement play a key role. Despite these advantages, most early research

focused on single-agent models that did not take into account the complexity of practical applications.

In multi-agent environments, numerous independent agents, which can represent different recommendation models, system components, or user profiles, simultaneously train and interact in a shared environment. Multi-agent systems are characterized by scalability, which allows agents to be changed or added without affecting the work of others, as well as stability and autonomy, which ensure their ability to work independently without human intervention. This creates additional challenges, in particular, the problem of environmental instability for each individual agent due to the constant updating of policies by other agents. In addition, ensuring effective coordination and information exchange between decentralized agents without compromising scalability and performance remains a difficult task.

Thus, developing and exploring multi-agent-based recommendation systems is a relevant and promising area that combines advances in machine learning and content personalization under conditions of growing data volumes and information load.

2. Literature review and problem statement

In [7], the results of research on the development of the multi-agent model Multi-Agent Collaboration Framework for Recommendation (MACRec) are reported. The model uses specialized agents (manager, analyst, reflector, searcher, task interpreter) with the support of a large language model (LLM) to perform a wide range of recommendation tasks. It is shown that the use of agents with different roles facilitates the effective resolution of complex tasks through distribution and adaptive interaction. However, issues related to the performance of the approach with limited computing resources or in real-time scenarios remain unresolved. The reason is the complexity of calculations since LLM agents require significant computational resources and memory. An option to overcome these difficulties may be the use of simpler models or hybrid configurations.

In [8], the multi-agent model Category-aware Dual-agent Reinforcement Learning (CADRL) is proposed, in which two agents interact: the first generates recommendations based on current knowledge, the second provides them to the user in an understandable form. It is shown that this approach increases trust in the system due to the transparency of decisions. At the same time, the question of the adaptability of such a system remains open: the two-agent model focuses primarily on explaining the results but does not take into account the diversity of interests of each user separately. This is due to the difficulty of simultaneously ensuring the clarity of decisions and the flexibility of adaptations to individual needs. A solution may be to expand the multi-agent architecture so that individual agents are responsible for specific users.

In [9], the MATCHA architecture was developed, in which several agents cooperate in a dialog system, performing various tasks – from processing requests and selecting options to explaining results and controlling their reliability. It was shown that such coordination between agents improves the quality of recommendations in dialog environments. However, here too, the issues of individualization remain unresolved: agents specialize by functional roles, not by users, which limits the system's ability to take into account the long-term interests of a particular person. A possible way to overcome this problem is to move from functional specialization of agents to user specialization where each agent learns based on the behavior of its user.

In [10], the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm is presented, which extends the Actor-Critic approach to the conditions of cooperative and competitive interaction environments. The approach uses the concept of centralized learning and decentralized execution, which allows agents to coordinate their parameters during training based on the actions of other agents, and demonstrate coordinated behavior. Such a structure solves the problem of instability in an environment of many agents and demonstrates high stability when performing various tasks. However, despite its effectiveness, MADDPG remains sensitive to scaling the number of agents, which limits its application in large systems. In addition, the need for centralized access to the actions of all agents in the training phase poses challenges for using this approach in privacy conditions, which is often encountered in real systems. The reason is the objective difficulties associated with the exponential growth in computational costs with a linear increase in the number of agents and the need for a significant amount of training data. An option to overcome these difficulties may be the use of distributed learning methods and model compression. This is the approach used in [11], but it also has its own limitations.

In [11], the Multi-Agent Recommender System (MARS) algorithm is considered and the results of research on various decentralized machine learning algorithms using communication between agents are reported. It is shown that adaptive information transfer between agents makes it possible to reduce the load and preserve the quality of collaborative learning and outperforms five existing models of recommender systems. However, each agent uses different machine learning algorithms, which complicates the architecture and performance. In addition, issues related to the sensitivity of the employed algorithms to delays that occur in real-world tasks remain unresolved. Furthermore, the aspect of personalization in the product trading system has not been addressed.

In [12], three decentralized Multi-Agent Natural Actor-Critic (MAN) algorithms are proposed that use natural gradients and function approximation to optimize a shared model without a central agent. The authors provide theoretical information on convergence and demonstrate the effectiveness of their methods in reducing network congestion in transport systems. At the same time, the practical implementation of MAN in complex dynamic environments with a high degree of stochasticity in user behavior still remains an open question. Moreover, the issue of agent personalization in MAN is not considered, which limits the scope of its application in personalized recommendation systems. The reason is the use of a significant amount of computing resources, making the algorithm unsuitable for large-scale deployment. An option to overcome the difficulties may be the use of approximation methods and stochastic estimates of the natural gradient. This is the approach used in [13], which reports the results of research on approximation methods for reducing computational complexity in multi-agent systems for the field of e-commerce. However, it does not completely solve the scalability problem, as issues related to error accumulation during approximation remain unsolved due to the inability to accurately reproduce the true gradient in large-scale systems.

In [14], the results of research on a multi-agent recommendation system in which responsibility is distributed

among agents for individual product categories without communication are given. It is shown that the absence of direct communication between agents reduces computational costs and facilitates scaling without significant loss of accuracy. However, issues related to the problem of initial recommendation for new users remain unresolved. The reason is the difficulties associated with the lack of sufficient information for the initial recommendation. An option to overcome the difficulties may be to combine decentralized methods with centralized mechanisms for policy initialization. A similar approach is used in [15], which presents the DANSER multi-agent system for streaming services. However, the high complexity of the architecture and the long duration of training complicate its use in real time.

All this suggests that it would be appropriate to conduct research aimed at searching for scalable, personalized, and user-oriented multi-agent algorithms for recommender systems. Notably, previous studies have not yet solved the problem of balance between personalized recommendations and information exchange within the entire system in a scalable and adaptive manner. This uncertainty emphasizes the presence of a general open question in the field. There is no integrated multi-agent recommender system based on the Actor-Critic model that is able to provide individual recommendations for each user and effectively solve the task of generating recommendations at the initial stage of use. Such a system should not depend on a large number of computational resources and make recommendations as quickly as possible.

3. The aim and objectives of the study

The aim of this study is to design a multi-agent adaptive recommendation system based on reinforcement learning to increase efficiency and accuracy in providing personalized recommendations. This approach aims to improve the accuracy of recommendations and ensure flexible consideration of individual user preferences. The expected outcome is an increase in the relevance of recommendations for individual users in a dynamic environment, as well as the potential for further integration of the developed model into real-world information systems.

To achieve the goal, the following tasks were set:

– to select an experimental data set and implement methods for constructing multidimensional vector representations of elements based on the unique characteristics of each object;

– to develop both centralized and decentralized reinforcement learning models for recommendation tasks based on the Actor-Critic, with the goal of conducting subsequent experimental comparison;

– to perform training and testing of the proposed multi-agent decentralized system in a simulated environment using the selected experimental data set;

– to compare effectiveness of the proposed model with a baseline single-agent approach by applying appropriate evaluation metrics.

4. The study materials and methods

The object of this study is the process of improving the efficiency and accuracy of delivering personalized recommendations to users in systems, that employ reinforcement learning. The principal hypothesis of the study assumes that the combined use of decentralized agents and a shared agent, trained using the Actor-Critic model and the deep deterministic policy algorithm could make it possible to increase the quality and adaptability of recommendations for each individual user.

The work assumes that user preferences can be effectively reflected in a multidimensional space and that recent interactions have a greater influence on the formation of recommendations. The study uses a simplification, according to which interactions between users and system elements are described by a single numerical vector that incorporates both the unique properties of the items and user's past actions.

A number of methods were used to conduct this study. In particular, the method of multiple correspondence analysis (MCA) [16] was applied to process categorical data, which makes it possible to display the relationships between discrete features in the form of multidimensional vector embeddings. MCA makes it possible to detect hidden dependencies in the input data and reduce redundancy. To reduce the dimensionality of multidimensional vectors formed during preprocessing, the probabilistic principal component analysis method (Probabilistic PCA, PPCA) [17] was applied. Unlike the classical method of principal component analysis, PPCA is based not only on linear transformation of data but also on constructing a latent model that takes into account the stochastic nature of observations. This approach makes it possible not only to extract components that explain the largest proportion of variance but also to correctly process data containing missing values or noise. The use of MCA and PPCA is justified by the fact that the first method is effective for analyzing categorical variables, while the second provides a more flexible and reliable representation of high-dimensional numerical vectors, minimizing information loss and improving model generalization.

For natural language processing, a modern transformer model, Robustly Optimized BERT Pretraining Approach (RoBERTa), was chosen [18]. The model was developed by Facebook AI as an improvement over Google's BERT model through the optimization of the pretraining process and efficient work with large amounts of text data. The use of this particular model is explained by its high quality in semantic text representation tasks, in which it demonstrates better accuracy compared to other modern models.

Three main metrics were used for the comparative analysis of centralized and decentralized models: Precision@k, Recall@k, and NDCG@k [19]. Precision@k determines the proportion of relevant elements among the first k recommended items and is defined by the following formula

$$\text{Precision@k} = \frac{1}{k}\sum_{i=1}^{k} rel(i). \quad (1)$$

In formula (1), $rel(i)=1$ if the i-th recommendation is relevant, and $rel(i)=0$ otherwise. The Recall@k metric reflects the system's ability to find all relevant elements among the top-k results and is defined by the following formula

$$\text{Recall@k} = \frac{\sum_{i=1}^{k} rel(i)}{rel_{count}}. \quad (2)$$

In formula (2), rel_{count} is the total number of relevant items for the user. The NDCG@k (normalized DCG) metric evaluates the quality of the ranking of recommended items, assigning greater weight to correctly identified elements placed at higher positions and is calculated as

$$\text{NDCG@k}=\frac{\sum_{i=1}^{k}\frac{rel(i)}{\log_2(i+1)}}{\sum_{i=1}^{\min(rel_{count},k)}\frac{1}{\log_2(i+1)}}. \quad (3)$$

The use of these metrics makes it possible to carry out a comprehensive assessment of the quality of recommendations. In particular, Precision@k evaluates the proportion of relevant elements among the top-k proposed ones and characterizes the accuracy of the selection. Recall@k calculates the total number of relevant elements and allows for an assessment of the completeness of coverage. NDCG@k takes into account not only the presence of relevant elements but also the order of their placement in the list, enabling an assessment of the elements ranking.

The experimental part of the study was performed in a specially created environment developed in the Python programming language, version 3.12 (USA). Reinforcement learning models, training and evaluation functions were implemented using the PyTorch library. Additional libraries were used to generate a vector representation of objects, calculate metrics, and build the environment itself, including scikit-learn, Prince, and SciPy. The matplotlib library was used to build plots. Actor-Critic neural networks were trained using the DDPG algorithm. A separate module was built for batch data processing for both training and testing.

The experiments were conducted on a personal computer with GPU acceleration. The hardware configuration included an eight-core Intel Core i7-9700K processor (USA), 32 GB of DDR4 RAM, and an NVIDIA GeForce GTX 1650 graphics card with 4 GB of GDDR5 memory. The main operating system is Ubuntu 24.04. All calculations were performed locally without involving cloud computing resources.

5. Research results regarding a multi-agent personalized recommendation system

5. 1. Experimental dataset and vector representation of elements

To evaluate the effectiveness of applying multi-agent reinforcement learning in an environment that is as close as possible to real-world recommendation systems, the MovieLens dataset [20] was used. This dataset is maintained by the GroupLens research group and is available in different variations depending on the number of user ratings: 100K (100,000), 1M (1,000,000), and 32M (32,000,000).

The dataset contains real user ratings from the website of the same name for films on a scale from 0 to 5. The last update of the dataset was published in April 2024. In addition to the ratings, detailed characteristics about each film need to be collected to build the corresponding vector representations, train, and verify the models. To this end, external services are integrated, such as the Internet Movie Database (IMdb), which is one of the most recognizable resources about films. In particular, the OMDb API and TMDb API application programming interfaces are used to collect data. The first contains detailed data from the IMdb website, and the second is an open interface to another popular film service called The Movie Database.

To uniquely identify each film in the set, the following attributes are used: director, year of release, budget, popularity metrics, ratings (IMDb, TMDb, Metascore, Rotten Tomatoes), plot, genres, spoken languages, countries of production, total revenue, and others. This variety of features allows the system to accurately distinguish objects with similar characteristics, which generally improves the quality and accuracy of recommendations.

To process data using neural networks, it is necessary to transform complex and diverse film features into numerical vectors, known as embeddings [21]. The purpose of this transformation is to preserve the unique characteristics of each object and at the same time allow the system to find elements with similar properties using comparative metrics, such as Euclidean distance, cosine similarity, and Manhattan distance. In this section, a methodology for constructing high-quality embeddings based on various properties and determining the optimal dimension for such a representation is described. Film properties can be conditionally divided into three main types:

1) numerical data (e.g., release year, ratings, number of votes, budget, revenue, duration, etc.);

2) categorical data (e.g., genres, languages spoken, countries of production, age rating, directors, etc.);

3) textual data (e.g., plot descriptions from IMDb and TMDb).

A total of 12 numerical features were obtained, including numerical ratings, number of votes, Rotten Tomatoes and Metacritic ratings, budget, revenue, and popularity metrics. These features are further encoded using PPCA to reduce dimensionality and preserve informativeness. The result of this process is a multidimensional vector representation of the numerical features.

Categorical features are processed using MCA. This transformation allows one to detect patterns among properties such as age rating, directors, genres, spoken languages, countries of production, and companies. The initial combination of categorical features forms a vector representation of nearly 400 features. To ensure integration with other feature types and reduce computational complexity, the MCA method is applied to these features, which reduces the dimensionality of the representation to smaller-sized vectors.

The modern RoBERTa transformer model is used to encode text data. Plot descriptions from IMDb and TMDb are concatenated, separating them with a space mark. After that, the merged text is fed into the RoBERTa model to generate a semantic vector representation.

In the final stage, all three vector representations generated from numeric, categorical, and text features are merged, and PPCA is applied again to reduce the dimensionality of the final vectors. An empirical evaluation was conducted using different dimensions of the vector representation, including 64, 128, 256, and 512. Among them, according to performance and benchmarking criteria, such as the t-distributed stochastic neighbor embedding (t-SNE) depicted in Fig. 1, embeddings with dimension 128 showed the best balance between computational efficiency and representation informativeness.

The increase in dimensionality leads to a significantly higher consumption of computer resources mentioned earlier. That is why the use of 128 dimensions is sufficient for accurate discrimination of films in a large data set. To compare two vectors with each other, this work uses the Euclidean distance as the basic metric for vector correlation. For practical implementation, a Python class was created to construct vector representations of elements.

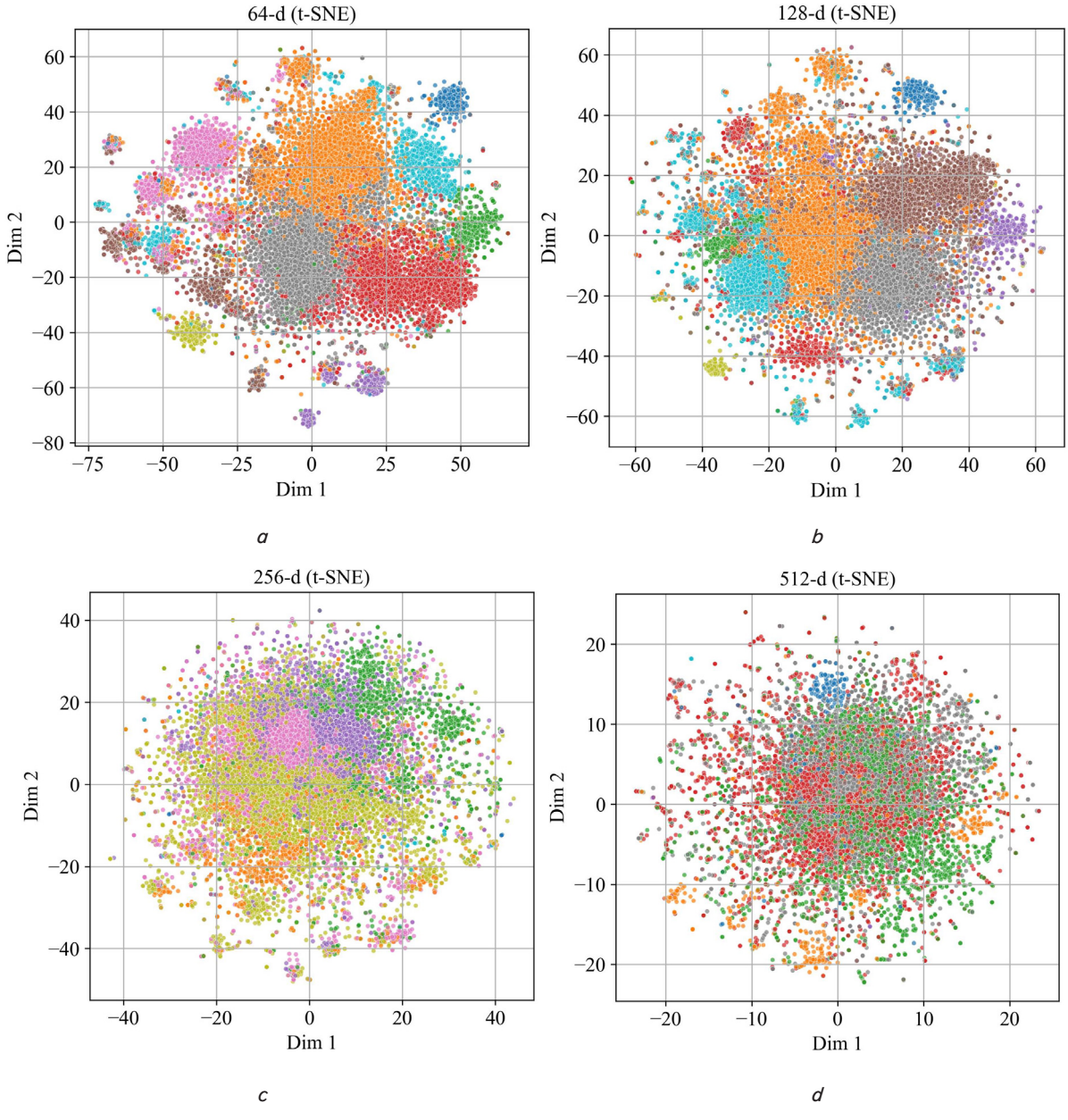


Fig. 1. t-SNE comparison of different embedding dimensions:
a – 64 features; *b* – 128 features; *c* – 256 features; *d* – 512 features

5. 2. Construction of a centralized and decentralized models of reinforcement learning

In this study, a model is proposed in which each user is assigned a separate independent agent, which makes it possible to minimize the influence of interactions of other users on the results of individual recommendations. Fig. 2 depicts the interaction between a user and their agent within a certain environment, modeled as a Markov decision process (MDP) [22].

Formally, an MDP is defined as a tuple of five elements (S, A, P, R, γ), where S is the state space, A denotes the set of actions, $P: S \times A \times S \to [0, 1]$ is the probability of state change, and $R: S \times A \times S \to \mathbb{R}$ is the reward function. The discount factor $\gamma \in [0, 1]$ determines the importance of short-term and long-term rewards: if $\gamma = 0$, then only short-term rewards are taken into account, and when $\gamma = 1$, long-term and immediate rewards have the same weight in calculating the benefit. At each step, the agent observes the current state of the user s_t and the reward r_t received for the previous interaction, and then generates the next action. Each agent receives user preference data and generates recommendations in order to maximize the immediate reward r_t, from the transition of the user's state from s_t to s_{t+1}. After receiving recommendations, the user provides feedback, for example, in the form of a rating, which is then used to update their current state.

For practical implementation, two separate environments were created: centralized (single-agent) and decentralized (multi-agent). Both environments are implemented as independent classes in the Python programming environment. Both implementations have common resources: access to user data, their ratings, metadata about films, vector representations, algorithms for calculating metrics, as well as mechanisms for training and generating recommendations.

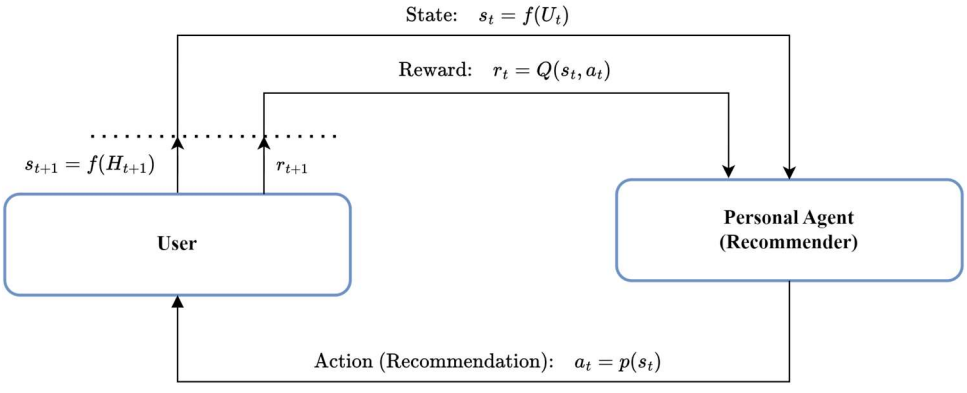


Fig. 2. Recommendation environment represented as an MDP

In a centralized environment, only one agent operates. It learns based on interactions with all users and forms recommendations using the accumulated common experience. The weights of the Actor and Critic neural networks in this case are stored in common files, ensuring a unified strategy for all users.

In a decentralized environment, there exists a set of agents, each corresponding to an individual user. The weights of the networks are stored in independent files tied to the unique identifier of a particular user. However, complete isolation of agents may limit access to common experience. To mitigate this problem, a shared agent was added. This central agent accumulates information from all users and helps overcome the "cold start" problem by providing relevant recommendations in the early stages of interaction. Personal agents in a decentralized environment are adapted versions of the shared agent. They are additionally trained on individual data of specific users.

In addition to the environments, Python classes were implemented for training neural networks in both centralized and decentralized models. Additional training mechanisms for personal agents are also provided. To compare the results, separate modules were built for plotting training loss graphs, as well as a simulator for comparative analysis of the operation of both environments. The result of the implementation is a recommendation system that includes two isolated environments: centralized and decentralized. Each of them provides agent training, recommendation generation, metric computation, and visualization of loss dynamics.

To build each individual agent in both environments, proposed system uses the popular Actor-Critic architectural model, which consists of two neural networks:

– Actor network. It is focused on learning a policy in order to generate recommendations that meet the user's current preferences. Its output is a vector representation that reflects the user's current needs as closely as possible;

– Critic network. It uses Q-learning to optimize the recommendation list generation by maximizing user engagement, and the output of this network is a scalar reward value.

The architecture of these networks, as shown in Fig. 3, includes a state representation module, several fully connected layers, a ranking procedure, and the use of Dropout mechanisms that randomly deactivate signals to reduce the risk of overfitting the model. The idea of this architecture was adapted from [23] with modifications that include adding extra fully connected layers, changing hyperparameters, and changing individual activation functions. The modifications to the architecture are depicted in Fig. 3 as orange blocks outlined by a dashed border. In Fig. 3, the input feature dimensionality for each fully connected layer is indicated on the left, while the output dimensionality is shown on the right.

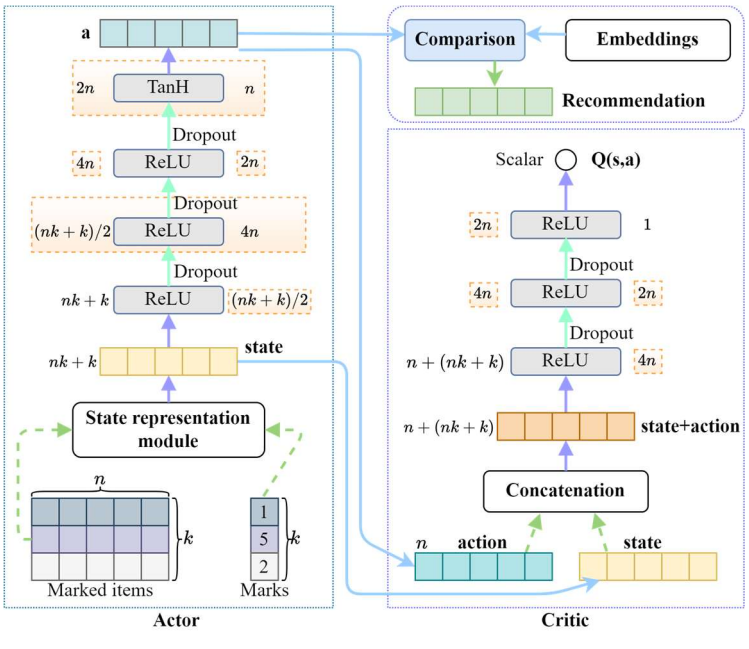


Fig. 3. Architecture of the Actor and Critic networks

The Actor network, also known as the policy network, is shown on the left in Fig. 3. For a given user, this network is responsible for generating action a based on current state s. Initially, the user state is formed using vector representations of the k most recent objects, with which the user has positively interacted, and which are the input data of the network. These vectors are processed by the state representation module, resulting in the formation of generalized state $s_t = f(U_t)$; in this study – through a simple concatenation of vectors and ratings. The generalized state s_t is input to three fully connected layers with ReLU (Rectified Linear Unit) activation. Next, this state passes through a fully connected layer with a hyperbolic tangent activation function, resulting in the formation of action $a_t = p(s_t)$, which is the final output

of the Actor network. The resulting action a is considered as a vector $a \in \mathbb{R}^{1\times n}$, which defines the ranking function. Based on the comparison of the calculated scores for all items in the dataset, the user is offered the top-k items with the highest scores according to the selected metric. When the agent offers an item and receives feedback, the state is updated to the next step $s_{t+1} = f(U_{t+1})$, and the process is repeated.

The Critic network, also known as the value network, is shown in the lower right part of Fig. 3. It is implemented in the form of a deep Q-network and uses a deep neural network to approximate the function $Q(s_t, a_t)$, dependent on the state and action, and known as the Q-function. This function evaluates the quality of the recommendation generated by the Actor network. Specifically, the Critic takes as input the user state s_t, generated by the state representation module, together with the action a_t generated by the Actor. This network returns a Q-value, represented as a scalar value. Based on the estimated Q-values, the parameters of the Actor network are updated in order to improve the quality of the generated actions. The goal of this process is to maximize the Q-value.

5. 3. Training procedure

The recommender system is trained using DDPG, which involves the construction of initial and target networks, ensuring a more stable model. This approach combines the advantages of reinforcement learning and deep learning methods, creating a scalable and adaptive system focused on individual user needs. In this approach, the policy (Actor) and value (Critic) networks are optimized simultaneously. The goal of training is to find an approximate recommendation that maximizes long-term user engagement and satisfaction. The training procedure varies depending on whether the agent is shared or personalized.

The initial Actor network is denoted as $\pi_\theta(s)$ and parameterized by vector θ; the initial Critic network is denoted as $Q_\phi(s,a)$ and parameterized by vector ϕ, where s is the state, a is the action. Accordingly, the target networks are denoted as $\pi_{\theta'}(s')$ for the target Actor and $Q_{\phi'}(s',\bullet)$ for the target Critic. The goal of the Critic network is to minimize the root mean square error between the predicted Q-value and the target Q-value calculated using the Bellman equation

$$L_{Critic}(\phi) = E_{(s,a,r,s')\sim D}\left[\left(Q_\phi(s,a) - y\right)^2\right]. \quad (4)$$

In formula (4), $y = r + \gamma Q_{\phi'}(s', \pi_{\theta'}(s'))$, r is the immediate reward, E is the expectation operator reflecting the mean value of the function over a given distribution. D is the experience replay buffer, $\gamma \in (0,1]$ is the discount factor, and ϕ', θ' are the parameters of the target networks, which are updated gradually using the soft update mechanism. The target networks stabilize learning due to the slow shifting of the target values in time-difference learning (TD-learning). The Actor is optimized using the gradient theorem of the deterministic policy. The gradient of the expected reward $J(\theta)$ is approximated as

$$\nabla_\theta J(\theta) \approx E_{S\sim D}\left[\nabla_a Q_\phi(s,a)\Big|_{a=\pi_\theta(s)} \cdot \nabla_\theta \pi_\theta(s)\right]. \quad (5)$$

The Actor network learns to choose actions that, according to the Critic's prediction, will lead to high rewards. In practice, the loss function for the Actor network is defined as the negative expected value of the Q-function with the current policy

$$L_{Actor}(\theta) = -E_{S\sim D}\left[Q_\phi(s,\pi_\theta(s))\right]. \quad (6)$$

Both networks are optimized using the Ranger optimizer, which combines Rectified Adam (RAdam) with Lookahead to improve stability and convergence. Dropout regularization with a probability of 10% is also used, which prevents overfitting by randomly deactivating neurons during training. For the centralized agent, the learning rate is set to $1\cdot10^{-5}$ for both Actor and Critic networks, the regularization coefficient is set to $1\cdot10^{-2}$ for the Critic and $1\cdot10^{-4}$ for the Actor. For the personalized model (user-oriented), the learning rate is increased to $1\cdot10^{-4}$ for both networks to strengthen the influence of gradients obtained from user data.

The gradient backpropagation process in the Actor network uses a gradient clipping in the range [–1, 1] to avoid the exploding gradient problem, which can make the training process unstable and divergent. The Critic's losses are calculated at each training step, while the Actor is updated less frequently (every 10–15 steps), taking into account the slower dynamics of policy update. The target networks $Q_{\phi'}$ and $\pi_{\theta'}$ are updated using soft updates, defined as

$$\tau\phi + (1-\tau)\phi' \to \phi', \tau\theta + (1-\tau)\theta' \to \theta'. \quad (7)$$

For the centralized model, $\tau = 1\cdot10^{-3}$ is set in formula (7), which allows the target networks to gradually converge with the main ones without destabilizing the learning process. For the decentralized model, this parameter in formula (7) is reduced to $\tau = 1\cdot10^{-4}$ to ensure greater stability in the training of the target networks. The specified hyperparameters (learning rate and τ) were selected empirically to ensure stable convergence in both the global and personalized learning models.

In the centralized architecture, the model is trained on batches of interactions (batch processing) selected from the general user population. Each batch includes data from several users (e.g., 25 users), which allows the model to learn a general behavioral strategy that reflects common behavioral patterns. This approach promotes generalization but may not take into account individual preferences.

In a decentralized architecture, the general model is further trained using interaction data from a specific user. The training process involves multiple passes over the same interaction history, which compensates for the data sparsity. To enhance the influence of this information, the loss function can be scaled, or the learning rate can be temporarily increased. In this study, the learning rate increase mentioned earlier is applied.

These two approaches reflect the core idea of the proposed model: centralized training provides scalability and consistency, while additional decentralized training provides personalization and local adaptation. Initializing personalized agents based on a shared model allows combining the advantages of both paradigms. It is worth noting that the Actor-Critic architecture is identical for both single-agent and multi-agent environments.

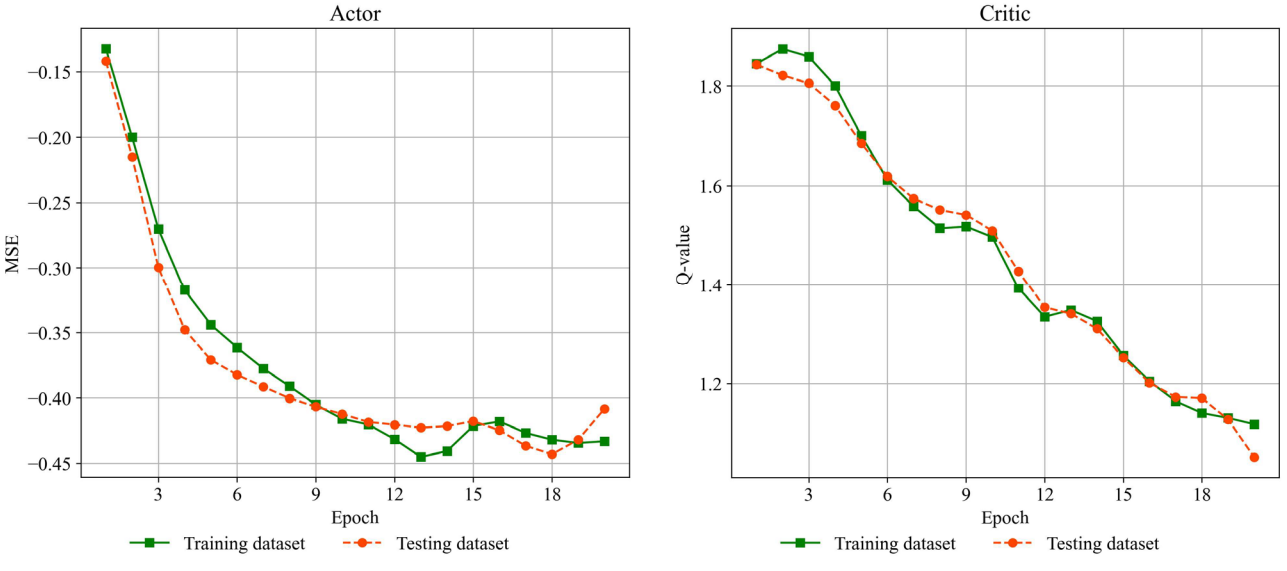
From the MovieLens 32M dataset, approximately 3.6 million ratings from 30,000 users were used. For this dataset, ratings greater than 2.5 are considered positive, while those less than 2.5 are considered negative. To construct the reward function, the ratings were empirically normalized to the range [–1, 1] and used as feedback from the received recommendation. Thus, the reward function can be defined by the following formula

$$R(s,a)=2\frac{r_{i,j}(t)}{5}-1. \quad (8)$$

In formula (8), $r_{i,j}(t)$ is the score from the interaction of user i with element j at time t.

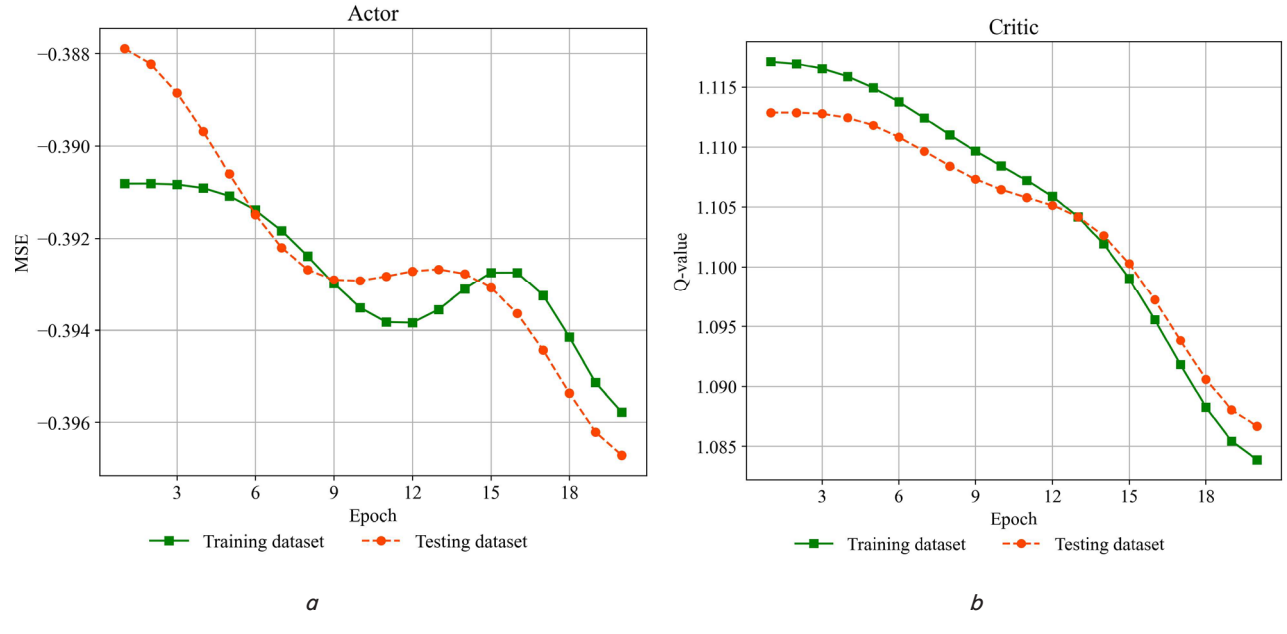
In the experimental study, 80% of all interactions (i.e.,data from 24,000 users, which is 3.34 million ratings) were used for training, and the remaining 20% (6,000 users, 330,000 scores) were used for testing. The sampling was done randomly. Additionally, 100 users were randomly selected from the test set, for whom individual agents were trained, enabling the simulation of a personalized environment. In such an environment, for each user, 70% of his interactions were used for training, and 30% for evaluating the decentralized model. The value of the discount factor in formula (4) $\gamma = 0.99$. The training of the shared agent was carried out for 20 epochs and lasted 9 minutes and 40 seconds. The personalized agents were trained for an additional 20 epochs in an average of 6 seconds. The policy and Q-values losses when training the Actor-Critic model using the shared agent are shown in Fig. 4. For comparison, the corresponding losses for the personalized agent are shown in Fig. 5.

Only offline simulations were conducted. The complete training of individual agents and the comparison of approaches across 100 test users took 5 minutes and 28 seconds. To store the networks weights of all these users within the simulation, an additional 700 MB of memory is required each time. The list of users for comparison is generated randomly at each simulation launch.



a *b*

Fig. 4. Training losses of the shared (central) agent: *a* – Actor losses; *b* – Critic losses



a *b*

Fig. 5. Training losses of a personal agent (user #17035): *a* – Actor losses; *b* – Critic losses

5. 4. Metrics evaluation

To compare the multi-agent and single-agent approaches, the metrics Precision@k, Recall@k, and NDCG@k were used as the basic metrics. To select the top-k elements, 5 and 10 were selected for comparative analysis. Tables 1, 2 give the results from the simulation, with the best values corresponding to the personalized model.

Table 1

Comparing metrics on a dataset of 5 elements

Model	Precision@5	Recall@5	NDCG@5
Central	0.81	0.1038	0.8173
Personalized	0.85	0.107	0.8465
Difference	+ 0.04	+ 0.0032	+ 0.0292

Table 2

Comparing metrics on a dataset of 10 elements

Model	Precision@10	Recall@10	NDCG@10
Central	0.847	0.2161	0.8487
Personalized	0.857	0.2179	0.8601
Difference	+ 0.01	+ 0.0018	+ 0.0114

Results indicate that the personalized multi-agent environment demonstrates higher efficiency compared to the centralized model in all metrics. In particular, the improvement in Precision@k and Recall@k indicates the ability of the decentralized model to more accurately identify relevant items. Higher values of NDCG@k indicate that relevant recommendations are more often at the top of the list. The results are confirmed by repeated simulations, and the multi-agent system consistently demonstrated an advantage. The data obtained demonstrates that the decentralized, user-oriented model not only finds appropriate content with greater accuracy but also assigns it higher priority.

In addition, a simulation was also conducted for a single user to directly compare the total reward received in centralized and decentralized environments. The previously calculated metrics Precision@k, Recall@k, and NDCG@k are focused on evaluating the relevance of items within a sample of multiple users. In contrast, the following comparison is based on the analysis of the rewards (ratings) received from a single user when applying the multi-agent and single-agent approaches. For this purpose, user number 17035 was randomly selected from among the users with the largest number of ratings. This user left a total of 4772 ratings. One recommendation of 10 items was generated for the current state of the user using both models, which made it possible to assess the difference in the quality of the recommendations. The results are given in Table 3.

Table 3

Comparative simulation for one user

No.	Recommendation (Central Agent)	Rating	Recommendation (Personalized)	Rating
1	Rich in Love (1992)	1	Rich in Love (1992)	1
2	The Man in the Moon (1991)	2	Underneath (1995)	3
3	The Siren (2019)	2.5	Between Us (2012)	3.5
4	Leap of Faith (1992)	3	Run with the Hunted (2019)	3
5	Boiling Point (1993)	0.5	Beautiful Girls (1996)	2
6	Ain't Them Bodies Saints (2013)	2	The Siren (2019)	2.5
7	Between Us (2012)	3.5	Don't Breathe 2 (2021)	3.5
8	Return to Paradise (1998)	2	Twins (1988)	3.5
9	Ghost (1990)	1	Bless the Child (2000)	2.5
10	Don't Breathe 2 (2021)	3.5	The Keep (1983)	2.0
Total reward		21	Total reward	26.5

It is important to emphasize that the multi-agent model orders recommendations more effectively, placing items with higher scores at the beginning of the list, which indicates its adaptability to user preferences and behavior, as it was additionally trained on the user's personal data. It is also noteworthy that the first recommended item coincides in both lists. The experiment was repeated 100 times for this user. In 74 cases, the decentralized model achieved a higher or equal total reward, of which in 66 cases it outperformed the centralized model for each individual recommendation, and in 8 cases both models gave the same result. Thus, the multi-agent system demonstrated a superior ability for personalized adaptation over time, generating more relevant and prioritized recommendations.

6. Discussion of results based on studying the multi-agent adaptive recommendation model

The results of this study demonstrate the advantages of using a multi-agent approach with the Actor-Critic architecture for building recommendation systems. The loss function plots shown in Fig. 4, 5 show stable convergence on both training and test data, which indicates effective matching of model parameters and the absence of overfitting. A smooth decrease in the loss values during the training process, as well as their stability on validation data, indicates effective generalization of models. In addition, this indicates the capability of the proposed architecture to adapt to new data in both centralized and decentralized approaches.

According to the comparison in Tables 1, 2, the selected architecture provides better results than the results of operating a single-agent architecture only in cases of a small number of elements that are allocated for generating a recommendation. In other cases, the use of the selected architecture does not provide significant advantages over the use of the conventional single-agent architecture. However, such scenarios with a limited number of recommended elements are the most typical for real-world applications, so the proposed approach has practical value. This improvement is explained by the decentralized structure of the multi-agent model, which allows each user to interact with a separate independent agent. Such a distribution reduces the influence of other users' behaviors, which is a known limitation of many centralized systems.

Unlike the multi-agent approaches considered in [7, 10, 12, 15], the proposed architecture is based on a simplified model that contributes to increased performance, optimal use of computing resources, and reduced memory costs. Unlike [9], in which the multi-agent system is organized by functional roles, the proposed architecture is focused on personalized interaction with individual users. Unlike a similar approach studied in [10], which focuses primarily on mixed cooperative-competitive tasks, the proposed model is focused on personalized recommendation scenarios with continuous individual adaptation.

Compared to [11], where each agent uses different machine learning algorithms, the proposed system provides a unified model with a single learning policy. This approach increases stability and reduces both training and recommendation generation time.

Compared to works [8, 12], which do not pay much attention to real-time personalization within a multi-agent environment, the proposed model gradually updates the user's policy according to new interactions. Unlike [12], the proposed model uses a shared agent. In addition, this study applies custom mechanism for constructing vector representations based on the unique features of elements. This allows the recommendation process to improve as the amount of data grows.

Compared to work [13], which focuses on the approximation of natural gradients, which leads to the accumulation of errors in large systems, the proposed architecture implements error control mechanisms and agent independence, which provides more stable learning and increased accuracy of recommendations.

Work [14], similar to the proposed model, ignores communication between agents to reduce the computational load, but this limits the system's ability to adapt to new users. Instead, proposed system integrates decentralized agents with centralized initialization mechanisms, which allows to effectively solve the problem of recommendation for new users and provides better adaptability.

Unlike [15], which uses complex architecture with a long training time, the proposed model has a simplified structure with a focus on efficiency and real-time application, which makes it more practical for large systems. In the proposed system, a recommendation of 10 items occurs in less than one second.

The results obtained, given in Table 3, demonstrate the adaptability of the proposed approach to individual user preferences and allow to partially resolve the problem that works [7–15] could not fully cover in terms of personalization and decentralization. The proposed approach provides an effective compromise between personalization and decentralization, which is confirmed by experiments with different learning policies and the use of independent agents. One hundred repetitions of the simulation for a random user showed a stable advantage of the multi-agent system in 66% of cases, which further confirms its effectiveness compared to the single-agent approach. Identical results in 8% of cases and worse in 26% of cases can be due to the adaptive behavior of the model.

Therefore, to solve the tasks of centralized and impersonal recommendations, this study has proposed a system based on multi-agent reinforcement learning. This makes it possible to model the recommendation process as a series of sequential decisions made by individual agents and makes the system more flexible and scalable. The proposed architecture uses the concept of centralized learning with decentralized execution, which allows agents to learn common behavioral patterns during the training process, while maintaining independence at the execution stage. This approach makes it possible to effectively take into account the dynamics of user preferences, optimize both short-term and long-term rewards, and ensure scalability in large interactive environments.

However, the current study has some limitations. The training of the models was carried out on a local computer with limited computational resources, which caused restrictions on the sample size and model complexity. In addition, only a part of the open data set was used, not the whole one. For better validation of the results and implementation in real applications, larger data sets and more powerful hardware may be required. The problem of adapting a decentralized approach to recommending popular system elements has not yet been considered in current study. This may be a direction for future research. In particular, through the development of agent interaction mechanisms or new hybrid centralized-decentralized approaches.

At the same time, the study also has certain drawbacks that could potentially limit the achievement of the expected results even within the specified conditions. In particular, in cases of short or weakly variable user interaction history, there is a risk of overfitting, which can reduce the quality of personalization and negatively affect the recommendation process itself. In addition, the existing model does not yet take into account contextual or individual factors, which are often critical in real dynamic environments, such as news platforms or social networks.

A promising area for further development of this work is the integration of the constructed model into an online platform in order to assess the benefits based on real interaction with users. It is important to note that it is also advisable to design a more robust user state representation module to better take into account the relationships between items and user behavior.

7. Conclusions

1. Within this study's context, an effective method for constructing vector representations based on unique properties of elements has been implemented. Different dimensionalities of vectors were tested in order to achieve a compromise between the complexity of the model and the feature expressiveness. The optimal dimension was found to be 128, which best reflects the semantic differences between objects. The proposed approach uses a combination of different methods to construct a vector representation, which makes it possible to place similar objects close to each other in embedding space. This approach not only improves the relevance of the recommendation compared to static encodings but also solves the problem of the weak representation of unique features.

2. Reinforcement learning models using single-agent and multi-agent approaches based on the Actor-Critic architecture have been developed. The multi-agent architecture involves the creation of a separate agent for each user, which allows for providing individualized recommendations focused on the preferences of a particular user. This approach eliminates the limitations inherent in other recommender systems considered in previous studies. Continuous updating of the Actor and Critic networks allows the system to dynamically adapt to changes in user behavior.

3. The proposed model was trained and validated in a simulated environment using the well-known MovieLens dataset. The training process demonstrated a stable improvement of policies and an increase in the total reward, which indicates the correct functioning of the simulated environment, in particular, the management of data processing, state updates, and recommendation selection. The achieved stability using conventional hardware confirms the practical feasibility of the proposed approach.

4. A comparative analysis of the trained decentralized model with a single-agent version was conducted using standard evaluation metrics. The user-oriented multi-agent model demonstrated higher performance for such metrics as Precision@5 (+ 4%), Recall@5 (+ 0.32%), and NDCG@5 (+ 2.92%) compared to the centralized implementation. When the list of recommendations was expanded to 10 items, improvements were also recorded: Precision@10 – by +1%, Recall@10 – by +0.18%, NDCG@10 – by +1.14%. In addition, a series of simulations were performed for an

individual user, in which the personalized agent was run 100 times and outperformed the shared agent in terms of cumulative reward in 66% of cases. Obtained results indicate that decentralized agents focused on an individual user are more effective in taking into account individual needs and are able to adapt over time, overcoming the limitations of static and centralized recommendation algorithms.

Conflicts of interest

The authors declare that they have no conflicts of interest in relation to the current study, including financial, personal, authorship, or any other, that could affect the study, as well as the results reported in this paper.

Funding

The study was conducted without financial support.

Data availability

All data are available, either in numerical or graphical form, in the main text of the manuscript.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the current work.

References

1. Zhang, Y. (2022). An Introduction to Matrix Factorization and Factorization Machines in Recommendation System, and Beyond. arXiv. https://doi.org/10.48550/arXiv.2203.11026
2. Wang, Y., Ren, Z., Sun, W., Yang, J., Liang, Z., Chen, X. et al. (2024). Content-Based Collaborative Generation for Recommender Systems. Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, 2420–2430. https://doi.org/10.1145/3627673.3679692
3. Saleh, A., Dharshinni, N., Perangin-Angin, D., Azmi, F., Sarif, M. I. (2023). Implementation of Recommendation Systems in Determining Learning Strategies Using the Naive Bayes Classifier Algorithm. Sinkron, 8 (1), 256–267. https://doi.org/10.33395/sinkron.v8i1.11954
4. Silva, N., Werneck, H., Silva, T., Pereira, A. C. M., Rocha, L. (2022). Multi-Armed Bandits in Recommendation Systems: A survey of the state-of-the-art and future directions. Expert Systems with Applications, 197, 116669. https://doi.org/10.1016/j.eswa.2022.116669
5. Liu, F., Tang, R., Li, X., Zhang, W., Ye, Y., Chen, H. et al. (2019). Deep Reinforcement Learning Based Recommendation with Explicit User-Item Interactions Modeling. arXiv. https://doi.org/10.48550/arXiv.1810.12027
6. Sumiea, E. H., AbdulKadir, S. J., Al-Selwi, S. M., Alqushaibi, A., Ragab, M. G., Fati, S. M., Alhussian, H. S. (2023). Deep Deterministic Policy Gradient Algorithm: A Systematic Review. https://doi.org/10.21203/rs.3.rs-3544387/v1
7. Wang, Z., Yu, Y., Zheng, W., Ma, W., Zhang, M. (2024). MACRec: A Multi-Agent Collaboration Framework for Recommendation. Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2760–2764. https://doi.org/10.1145/3626772.3657669
8. Zheng, S., Yin, H., Chen, T., Kong, X., Hou, J., Zhao, P. (2025). CADRL: Category-Aware Dual-Agent Reinforcement Learning for Explainable Recommendations over Knowledge Graphs. 2025 IEEE 41st International Conference on Data Engineering (ICDE), 128–141. https://doi.org/10.1109/icde65448.2025.00017
9. Hui, Z., Wei, X., Jiang, Y., Gao, K., Wang, C., Ong, F. et al. (2025). MATCHA: Can Multi-Agent Collaboration Build a Trustworthy Conversational Recommender? arXiv. https://doi.org/10.48550/arXiv.2504.20094
10. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., Mordatch, I. (2017). Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. arXiv. https://doi.org/10.48550/arXiv.1706.02275
11. Alhejaili, A., Fatima, S. (2021). Multi-Agent Recommender System. Recent Advances in Agent-Based Negotiation, 103–119. https://doi.org/10.1007/978-981-16-0471-3_7
12. Trivedi, P., Hemachandra, N. (2022). Multi-Agent Natural Actor-Critic Reinforcement Learning Algorithms. Dynamic Games and Applications. https://doi.org/10.1007/s13235-022-00449-9
13. Vullam, N., Vellela, S. S., B, V. R., Rao, M. V., SK, K. B., D, R. (2023). Multi-Agent Personalized Recommendation System in E-Commerce based on User. 2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC), 1194–1199. https://doi.org/10.1109/icaaic56838.2023.10140756
14. He, X., An, B., Li, Y., Chen, H., Wang, R., Wang, X. et al. (2020). Learning to Collaborate in Multi-Module Recommendation via Multi-Agent Reinforcement Learning without Communication. Fourteenth ACM Conference on Recommender Systems, 210–219. https://doi.org/10.1145/3383313.3412233
15. Wu, Q., Zhang, H., Gao, X., He, P., Weng, P., Gao, H., Chen, G. (2019). Dual Graph Attention Networks for Deep Latent Representation of Multifaceted Social Effects in Recommender Systems. The World Wide Web Conference, 2091–2102. https://doi.org/10.1145/3308558.3313442
16. Khangar, N., Kamalja, K. (2017). Multiple Correspondence Analysis and Its Applications. Electronic Journal of Applied Statistical Analysis, 10 (2), 432–462. Available at: https://www.researchgate.net/publication/320694285
17. Ghojogh, B., Ghodsi, A., Karray, F., Crowley, M. (2022). Factor Analysis, Probabilistic Principal Component Analysis, Variational Inference, and Variational Autoencoder: Tutorial and Survey. arXiv. https://doi.org/10.48550/arXiv.2101.00734

18. Pookduang, P., Klangbunrueang, R., Chansanam, W., Lunrasri, T. (2025). Advancing Sentiment Analysis: Evaluating RoBERTa against Traditional and Deep Learning Models. Engineering, Technology & Applied Science Research, 15 (1), 20167–20174. https://doi.org/10.48084/etasr.9703
19. Jadon, A., Patil, A. (2024). A Comprehensive Survey of Evaluation Techniques for Recommendation Systems. Computation of Artificial Intelligence and Machine Learning, 281–304. https://doi.org/10.1007/978-3-031-71484-9_25
20. Harper, F. M., Konstan, J. A. (2015). The MovieLens Datasets. ACM Transactions on Interactive Intelligent Systems, 5 (4), 1–19. https://doi.org/10.1145/2827872
21. Zhao, X., Wang, M., Zhao, X., Li, J., Zhou, S., Yin, D. et al. (2023). Embedding in Recommender Systems: A Survey. arXiv. https://doi.org/10.48550/arXiv.2310.18608
22. Leon, V., Etesami, S. R. (2023). Online Reinforcement Learning in Markov Decision Process Using Linear Programming. 2023 62nd IEEE Conference on Decision and Control (CDC), 1973–1978. https://doi.org/10.1109/cdc49753.2023.10383839
23. Romaniuk, B., Peliushkevych, O., Shcherbyna, Y. (2021). Recommendation system development using reinforcement learning. Visnyk of the Lviv University. Series Applied Mathematics and Computer Science, 29, 150–162. https://doi.org/10.30970/vam.2021.29.11016