UDC 004.855.5(045)
DOI:10.18372/1990-5548.76.17668

[1]V. M. Sineglazov,
[2]A. V. Sheruda

RECOMMENDER SYSTEMS BASED ON REINFORCED LEARNING

[1]Aviation Computer-Integrated Complexes Department, Faculty of Air Navigation Electronics and Telecommunications, National Aviation University, Kyiv, Ukraine
[2]Faculty of Informatics and Computer Science, National Technical University of Ukraine "Ihor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine
E-mails: [1]svm@nau.edu.ua ORCID 0000-0002-3297-9060, [2]sheruda.andrew@lll.kpi.ua

***Abstract*—*This article is devoted to the problem of building recommender systems based on the use of artificial intelligence methods. The paper analyzes the algorithms of recommender systems. Analyzes the Markov decision-making process in the context of recommender systems. Approaches to the adaptation of reinforcement learning algorithms to the task of recommendations (transition from the task of supervised learning to the task of reinforcement learning) are considered. Reinforcement learning algorithms Deep Deterministic Policy Gradient and Twin Delayed DDPG were implemented with their own environment simulating the user's reaction, and the results were compared. The structure of a recommender system has been developed, in which the recommender agent generates a list of offers for an individual user, using his previous history of ratings. In the system itself, the user has the ability to interact only with the space of recommended films. This can be compared to the main YouTube page, which is a feed with suggestions, but we have a user interacting only with this feed and his reaction to objects in the recommendation space falls into recommender agent, which regulates the parameters of the model in the learning process.*

Index Terms—Machine learning; reinforcement learning; recommendation systems; recommender agent; collaborative filtering; Actor-Critic; explicit feedback.

I. INTRODUCTION

Like any field of research, recommender systems are constantly evolving over the years, reaching new heights in the selection of information according to user preferences. As the complexity of algorithms and approaches grows, the standards of recommender systems are also increasing. From collaborative filtering and Pearson's correlation coefficient to powerful reinforcement learning models, recommender systems have come a long way in development and improvement. Modern recommender systems are able to create new sets of user preferences in real time, guided not only by their previous actions, but also taking into account any additional, and seemingly unimportant information, with the help of efficient model training solutions and built-in powerful neural networks.

Recommender systems are making a significant impact on the global economy, finding their purpose in every area where there is a relationship between objects and users. From movie recommendations on movie websites to investment recommendations for large businesses, recommender systems can be a game changer for any service in which they are implemented. In today's world, large amounts of information do not make it possible to independently assess the needs of an individual user, and irrelevant offers from a service or company can reduce the interest and activity of the audience to which they provide goods or services.

When the question of how to increase demand for products comes up, manufacturers pay attention not to the price-quality ratio, but to whether they have chosen the right audience for its distribution. Powerful recommendation systems can increase the service's profit without any price manipulation, but simply by offering certain products only to those who are really interested in them. Each user is not unique; they can be grouped by criteria, analyzed by their behavior, and predicted what they will like, which is the task of recommender systems.

Although the context of this work is movie recommendations, this does not place any serious limitations on the approach demonstrated. The dataset used, movielens-100k, was chosen only because of its popularity and prevalence, which makes it possible to objectively evaluate the results. The approach put forward in Section 3 can be used for any recommendation tasks and demonstrate the same high results.

II. RECOMMENDER SYSTEMS AND THEIR PROBLEMS

Driven by the explosive growth in the amount of digital information available and the number of Internet users, the problem of information overload

prevents timely access to information of interest to a particular user. This has led to an increase in demand for recommender systems more than ever before.

Recommender systems are a subclass of information filtering systems that solve the problem of information overload [1] by filtering out from a large amount of dynamically generated data only those fragments that are hypothetically interesting to the user, referring to his past preferences, history of interaction with the source and other user data. A recommender system is able to predict whether a particular user will prefer a certain piece of information (goods, services, videos, etc.) or not based on the user's profile data.

Recommender systems are extremely beneficial for both service users and service providers [2]. They reduce transaction costs for searching and selecting products in the online shopping environment [3]. It has also been proven that recommender systems improve decision-making and decision quality [4]. In e-commerce, recommender systems can increase revenues because they help users choose the products they are interested in [2]. Thus, the need to use effective and accurate recommendation methods in a system that will provide relevant and reliable recommendations to users cannot be overemphasized. Let's consider the challenges of recommender systems.

A. Collecting feedback from users

It is a fact that most users do not give any assessment of the product or the information presented. Thus, a research problem arises: how to find out how satisfied the user is with the product. There are two ways to get evaluations. The first one is an explicit evaluation of the user after they have purchased a product or after viewing information. The other way is to predict their ratings for a particular product based on their preferences for other products. This method is known as the implicit rating collection method. [5], [6].

B. The cold start problem

The cold start problem is the problem of lack of information about a user or object in the system. [7] In this case, recommendation systems based on collaborative filtering, which require mandatory information about the user or object before making a recommendation, lose their effectiveness. The cold start can be broken down into three sub-problems.

The first problem is that we don't have any information about a new user. For example, when they have just joined the system. This problem is known as the new user cold start problem [12].

The second problem occurs when we introduce a new item into the system, but this item is unique in its kind and the system cannot find any evaluation related to this object. For example, collaborative filtering systems that need a matrix of user ratings to make a recommendation will not be able to run. This problem is known as the item cold start problem. The third problem

The third problem occurred when we launched the system for the first time. In this case, we have neither user nor product information. In other words, we don't have a user-item rating matrix, which is necessary for recommender systems to work properly and collaborate properly. This problem is known as a system cold start problem. For cold start problems, well-known content-based solutions can be applied[8, 9] and combinations of different machine learning methods can be used [10], [11].

C. Sparsity problem

A common problem with recommender systems (RS) is that users do not provide adequate feedback. Thus, even though there may be many users in our system, it is quite possible that we will receive very few ratings from them. Sometimes users provide information noise, irrelevant ratings, and misclick results to the system. This is perceived by the recommendation system as input, which subsequently shows the user undesirable results, which spoils the platform's performance. Also, users rarely rate the entire set of objects on the platform, which leads to the same problem [15].

The sparsity problem is a domain problem of collaborative filtering algorithms in RS that arises due to the sparsity of the rating matrix. From a mathematical point of view, when the user-item rating matrix becomes sparse, it gives rise to a unique problem that is known as the sparsity problem in RS [13], [14].

D. Scalability

Scalability is a system property that determines how a recommender system copes with the growing number of objects and their properties in it [16].

Scalability issues can be divided into two parts: hardware scalability and software scalability. Hardware scalability is about increasing the power or amount of hardware to solve a scalability problem.

For example, you can increase the CPU, RAM, and server configuration to solve the problem. But only a hardware increase in bandwidth cannot solve the problem [16].

Software scalability is the ability of RS algorithms and methods to cope with the increasing number of

objects and their properties in the system. This is a major problem that is not as simple as it seems. Most algorithms demonstrate good results only in spaces with a small dimensionality. In this case, the prediction accuracy decreases with increasing data volume or their execution time increases to unacceptable intervals. Such algorithms are unable to keep pace with the growth of the platform and, therefore, the problem of scalability arises [13].

E. The problem of over-specialization

This problem occurs if the recommended objects are too similar to each other and as a result, RS provides the user with the same recommendations.

One solution to this problem is to diversify recommendations. In this case, we list all products that are not similar to each other but may be of interest to the user [17].

F. Lack of data

Perhaps the biggest problem faced by recommender systems is that they need a lot of data to provide recommendations effectively. In the world of recommender systems, it is a common practice to use publicly available datasets from another environment to evaluate the effectiveness of recommendation algorithms [18]. These datasets are very useful and are used as a benchmark for developing new recommendation algorithms. For all RS algorithms, the more data we have about the user and objects on the platform, the more relevant the recommendations will be.

G. Changing data

The next major challenge in the recommendation system is changing user preferences and keeping up with preferences that change too quickly. The user's intentions for viewing a particular item may be different at different times, so recommender systems that are entirely based on user preferences may provide incorrect recommendations [19].

III. Problem Statement

The general approach to recommender systems is formulated as follows:

Let the system have objects and users. The two entities are connected by the ratings that users give to an object. In the context of our problem, a rating is a positive integer from 1 to 5. Then all ratings can be represented as a matrix (Fig. 1).

Objects

Users	1	2	...	*i*	...	*m*
1		5		2	1	
2			4			
:		2			1	
j	4		2	?		4
:		1	2		3	
n		5			1	

Fig. 1. The matrix of estimates

Let there be a user *j*. The task is to predict how user j would rate object *i*. In this case, depending on the approach used, both the user and the object may have certain properties, and the evaluation may be any action.

There are usually four approaches to recommender systems [20]:

- content-based recommendations;
- collaborative filtering;
- demographic filtering;
- hybrid approaches.

A. Content-based recommendations

For each user *j*, we want the algorithm to predict the rating of object *i* using the function

$$\left(\theta^{(j)}\right)^{\mathrm{T}}\left(x^{(i)}\right),$$

where $\theta^{(j)}$ is a vector of parameters $\mathbb{R}^{n+1}$ for user *j*; $x^{(i)}$ is a vector of features $\mathbb{R}^{n+1}$ for object *i*; $y^{(i,j)}$ is the real evaluation of object *i* by user *j*.

The task is to minimize:

$$J\left(\theta^{(1)},\ldots,\theta^{(n_u)}\right)=\frac{1}{2n_u}\sum_{j=1}^{n_u}\sum_{i:r(i,j)=1}\left(\left(\theta^{(j)}\right)^{\mathrm{T}}\left(x^{(i)}\right)-y^{(i,j)}\right)^2+\frac{\lambda}{2n_u}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(\theta_k^{(j)}\right)^2.$$

Content-based filtering methods are based on the description of the object and the user's taste profile, and they place more emphasis on analyzing object attributes to make predictions. Recommendations are made based on user profiles using features extracted from the content of objects that the user has rated in the past [21], [22].

The user is recommended items that are mainly related to positively rated items. Content-based filtering uses different types of models to find similarities between documents to generate meaningful recommendations. It can use vector-space models such as term frequency inverse document frequency (TF/IDF) or probabilistic

models such as naive Bayes classifier [23], decision trees [24], or neural networks to model the relationships between objects. The content-based filtering technique does not require the profiles of other users in the system, as they do not affect the recommendations. In addition, if a user's profile changes, this method still has the potential to adjust its recommendations within a very short period of time. The main disadvantage of this method is the need for a wide range of parameters to describe objects for good performance.

B. Collaborative filtering

Given the values of $\theta^{(1)},\ldots,\theta^{(n_u)}$, minimize the square of the difference between the predicted ratings $\left(\theta^{(j)}\right)^{\mathrm{T}}\left(x^{(i)}\right)$ and the actual ratings $y^{(i,j)}$:

$$\min_{x^{(1)},\ldots,x^{(n_u)}} \frac{1}{2n_u}\sum_{j=1}^{n_u}\sum_{i:r(i,j)=1}\left(\left(\theta^{(j)}\right)^{\mathrm{T}}\left(x^{(i)}\right)-y^{(i,j)}\right)^2+\frac{\lambda}{2n_u}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(\left(x_k^{(j)}\right)^2\right)^2.$$

The collaborative filtering method is based on a matrix of user-object relationships and ratings. The system selects users with similar interests and preferences by calculating the similarity between their interaction histories in the system, and then provides recommendations [25]. Such users form a group called neighbors. The user receives recommendations for those products that he or she has not rated before, but which have already been rated by neighboring users.

C. Demographic filtering

The optimization task for demographic filtering can be formulated as follows. Minimize the cost function $J(\theta,x)$, where θ are model parameters corresponding to demographic categories; x is the user preference vectors; n_u is the number of users; *r(i, j)* is an indicator of whether user *i* has already expressed their preference for category *j*; $d^{(i)}$ are demographic characteristics of the user *i*; $y^{(i,j)}$ is an assessment of user *i*'s preference for category *j*.

The cost function is as follows:

$$J(\theta,x)=\frac{1}{2n_u}\sum_{j=1}^{n_u}\sum_{j:r(i,j)=1}\left(\left(\theta^{(j)}\right)^{\mathrm{T}}\left(x^{(i)}\circ d^{(i)}\right)-y^{(i,j)}\right)^2+\frac{\lambda}{2n_u}\sum_{i=1}^{n_u}\sum_{k=1}^{n}\left(x_k^{(j)}\right)^2.$$

Recommender systems based on demographic filtering (DF) classify users based on their demographic information and recommend services according to the type of user. In demographic filtering, user profiles are created by classifying users into stereotypical descriptions that represent the characteristics of user groups [26]. Demographic information identifies those users who are interested in similar services. DF creates categories of users with similar demographic characteristics, and then tracks the accumulated behavior or preferences of users in these categories. For a new user, recommendations are generated by first determining which category they belong to and then applying the accumulated preferences of previous users to that category. Similar to collaborative techniques, demographic techniques also generate human-to-human correlations, but use different data. Collaborative and content-based techniques require a history of user ratings, which is not necessary for a demographic-based approach.

D. Comparison of approaches

The use of efficient and accurate recommendation methods is a key factor for a productive recommender system. Approaches and algorithms may differ for different tasks, but it is important to evaluate all the benefits they can provide and the limitations that accompany them before choosing one.

As we can see in Table I, the choice of approach requires some compromises and therefore, when creating a recommender system, it is important to evaluate the purposes for which it will be created and the environment in which it will operate, i.e. what properties the users and objects in it will have.

In the broader context of recommender systems, there are also many other approaches and their variations that have their own advantages, for example:

- *Matrix factorization*: The approach is based on decomposing the user and item interaction matrix into smaller matrices, which allows finding hidden dependencies and making recommendations based on them.
- *Knowledge-based*: This approach uses expert knowledge or rules to make recommendations. It takes into account user requirements and item characteristics to make recommendations.
- *Transformer-based*: This approach uses a transformer model architecture for recommender systems. It is able to model long-term dependencies in the sequences of user interactions with the system

and items, which allows for more accurate and contextualized recommendations.

- *Reinforcement Learning-based*: This approach uses reinforcement learning techniques to build an optimal recommendation strategy. The system interacts with the environment, observes how users react to recommendations, and learns to improve recommendations based on the rewards they receive.

TABLE I. ADVANTAGES AND DISADVANTAGES OF RECOMMENDER SYSTEM APPROACHES

No	Advantages	Disadvantages	Techniques
1	1. The system does not use user data for recommendations.	1. It requires analyzing and determining all characteristics of items to create a recommendation list.	Content-based recommendations
	2. The system can recommend new items to users based on the similarity of their characteristics.	2. The system is not dependent on user ratings for a particular item, so the quality rating of a product is not considered.	
2	1. The system does not use demographic information for recommendations.	1. The quality of the system depends on the list of items with the highest ratings.	Collaborative filtering
	2. The system compares similar items among users.	2. There is a problem with how to recommend items to a new user (cold-start problem).	
	3. The system can recommend items that do not match the user's preferences, but they might still like them.		
3	1. The system is not based on user ratings of items; it provides recommendations before the user has rated any item.	1. Collecting demographic data raises confidentiality issues.	Demographic filtering
		2. Stability problem versus plasticity.	
4	1. Combines all the advantages of content-based and collaborative approaches.	1. There is a cold-start problem.	Hybrid approaches
	2. Based on content description and user ratings.	2. Early adopter problem for products.	
	3. Helps avoid recommendation saturation.	3. Data sparsity problem.	

Our task is to consider and implement the latter approach.

IV. ALGORITHMS

A. Deep Deterministic Policy Gradient

The task is to minimize:

$$J(\theta'_\mu, \theta_\mu) = \left(r + \gamma Q'\left(s', a'; \theta'_\mu\right) - Q\left(s, a; \theta_\mu\right)\right)^2,$$

where θ_μ are critic network parameters; θ'_μ are parameters of the target Critic network; $Q\left(s, a|\theta_\mu\right)$ is the critic network result for (*s, a*); $Q'\left(s', a'; \theta'_\mu\right)$ is the target Critic network result.

Algorithm

1: Initialize the Actor f_{θ_π} and Critic $Q\left(s, a|\theta_\mu\right)$ networks with random weights.

2: Initialize the target networks f' and Q' with weights $\theta'_\pi \leftarrow \theta_\pi, \quad \theta'_\mu \leftarrow \theta_\mu$.

3: Initialize replay buffer *M*

4: for session = 1 to *M* do

5: Refresh space of items *I*

6: Initialize state s_0 from previous sessions

7: for t = 1 to *T* do

8: Stage 1: Transition generating stage

9: Select an action a_t according to f_{θ_π}

10: Execute action a_t and observe the reward r_t

11: Save transition $\left(s_t, a_t, r_t, s_{t+1}\right)$ in *M*

12: $s_t \leftarrow s_{t+1}$

13: Stage 2: Parameter updating stage

14: Sample minibatch with *N* transitions $\left(s, a, r, s'\right)$ from *M*

15: *Generate* a' according to target Actor network f_{θ_π}

16: $y \leftarrow r + \gamma Q'\left(s', a'; \theta'_\mu\right)$

17: Update Critic by minimizing

$\left(y - Q\left(s, a; \theta_\mu\right)\right)^2$ according to:

$$\nabla_{\theta_\mu} L\left(\theta_\mu\right) \approx \frac{1}{N}\sum_i \left(y - Q\left(s, a; \theta_\mu\right)\right)\nabla_{\theta_\mu} Q\left(s, a; \theta_\mu\right)$$

18: Update Actor using the sampled policy gradient:

$$\nabla_{\theta_\pi} f_{\theta_\pi} \approx \frac{1}{N}\sum_i \nabla_a Q\left(s, a|\theta_\mu\right)\nabla_{\theta_\pi} f_{\theta_\pi}\left(s\right)$$

19: Update the Critic target network:

$$\theta'_\mu \leftarrow \tau\theta_\mu + \left(1-\tau\right)\theta'_\mu$$

20: *Update the Actor target network:*

$$\theta'_\pi \leftarrow \tau\theta_\pi + \left(1-\tau\right)\theta'_\pi$$

21:end for

22: end for

B. Twin Delayed DDPG

Twin Delayed DDPG (TD3) is a modification of the Deep Deterministic Policy Gradient (DDPG) algorithm that uses several innovations. Smoothing the target policy:

$$a'\left(s'\right) = \text{clip}\left(f'_{\theta_\pi}\left(s'\right) + \text{clip}\left(\in, -c, c\right), a_{\text{Low}}, a_{\text{High}}\right),$$

$\in \sim \mathcal{N}\left(0, \sigma\right)$ – noise.

$a_{\text{Low}} < a < a_{\text{High}}$ is the clip range of the target action.

Smoothing the target policy serves to regularize the algorithm. This regularization option is intended for a specific type of biases when $Q\left(s, a; \theta_\mu\right)$ forms an incorrect sharp peak for some actions, the policy may behave erratically because of this.

The TD3 algorithm is based on two different Critics $Q_{\phi_i}, i = 1,2$, each of which has the same properties. A similar solution is also created for the purpose of regularization. The target value will have the form:

$$y\left(r, s', d\right) = r + \gamma \min_{i=1,2} Q_i\left(s', a'\left(s'\right), \theta'_{\mu,i}\right).$$

TD3 updates the policy only with a certain frequency, which contributes to the stable operation of the algorithm. Typically, the policy is updated every second iteration.

The task is to minimize:

$$J\left(\theta'_\mu, \theta_\mu\right) = \left(r + \gamma \min_{i=1,2} Q'_i\left(s', a'(s'), \theta'_{\mu,i}\right) - Q_i\left(s, a; \theta'_{\mu,i}\right)\right)^2 \quad i = 1, 2$$

where θ_μ are parameters of Critic network; θ'_μ are *parameters of Critic target network*; $Q_i\left(s, a|\theta_\mu\right)$ is the result of the work of the Critic's network i for (s, a); $Q'_i\left(s', a'; \theta'_\mu\right)$ is the result of the work of the Critic's target network i for $\left(s', a'\right)$.

Algorithm

1: Initialize Actor network f_{θ_π} and critic's $Q_1\left(s, a|\theta_{\mu,1}\right)$ and $Q_2\left(s, a|\theta_{\mu,2}\right)$ with random weights.

2: Initialize target networks f'_1 and Q'_i with weights

$\theta'_\pi \leftarrow \theta_\pi, \quad \theta'_\mu \leftarrow \theta_\mu$.

3: Initialize replay buffer M

4: for session = 1 to M do

5: Refresh space of items I

6: Initialize state s_0 from previous sessions

7: for t = 1 to T do

8: Stage 1: Transition generating stage

9: Select an action a_t according to f_{θ_π}

10: Execute action a_t and observe the reward r_t

11: Save transition $\left(s_t, a_t, r_t, s_{t+1}\right)$ in M

12: $s_t \leftarrow s_{t+1}$

13: Stage 2: Parameter updating stage

14: Sample minibatch with N transitions $\left(s, a, r, s'\right)$ from M

15: *Generate* a' according to target Actor network f_{θ_π}

16: $y \leftarrow r + \gamma \min_{i=1,2} Q_i\left(s', a'\left(s'\right), \theta'_{\mu,i}\right)$

17: Update Critic by minimizing

$\left(y - Q_i\left(s, a; \theta_{\mu,i}\right)\right)^2$ according to:

$$\nabla_{\theta_{\mu,i}} L\left(\theta_{\mu,i}\right) \approx \frac{1}{N}\sum_i \left(y - Q_i\left(s, a; \theta_{\mu,i}\right)\right)\nabla_{\theta_{\mu,i}} Q_i\left(s, a; \theta_{\mu,i}\right), \quad i=1,2$$

18: Update Actor using the sampled policy gradient,

if session % 2=0:

$$\nabla_{\theta_\pi} f_{\theta_\pi} \approx \frac{1}{N}\sum_i \nabla_a Q_1\left(s, a|\theta_\mu\right)\nabla_{\theta_\pi} f_{\theta_\pi}\left(s\right)$$

19: Update the Critic's target network:

$$\theta'_{\mu,i} \leftarrow \tau\theta_{\mu,i} + \left(1-\tau\right)\theta'_{\mu,i}, \quad i = 1,2$$

20: *Update the Actor target network:*

$$\theta'_\pi \leftarrow \tau\theta_\pi + \left(1-\tau\right)\theta'_\pi$$

21:end for

22: end for

V. RESULTS OF WORK

The results were obtained using the movielens-100k dataset.

This dataset represents the history of ratings of 1682 movies by 943 users, with a total of 100,000 records (Fig. 2).

```
(100000, 4)
```

	user_id	movie_id	rating	unix_timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596

Fig. 2. Data representation in movielens-100k

The reason for its selection is that this dataset is quite popular among developers of recommender systems and is usually used for testing RS.

To evaluate normalized discounted cumulative gain (NDCG) and mean reciprocal rank (MRR), we use the relevance metric – the score multiplied by 0.2. That is, if the rating is 5, then the relevance is maximum, and so on. For hit rate (HR), a score of 4 or 5 is considered the correct answer [27].

Usually, in the process of training Reinforcement learning (RL) models, a graph of cumulative reward growth over an epoch is plotted to visualize the improvement of the model. However, because our model uses a replay buffer, the reward is averaged and we cannot track it for each individual trajectory. In fact, our model does not feed the data of an individual user sequentially and does not calculate the cumulative reward for the user over the entire trajectory of his or her history, but we build a trajectory for all users simultaneously. In the context of RS, this feature is a significant advantage without which the system implementation is not possible. In general, to evaluate the quality of the system, it is enough to use the relevance metrics of recommendations and the distribution of ratings of movies that RA has recommended to the entire user population.

To get better results, the model should be run 4–5 times on training data, because simply increasing the number of epochs is not enough. The reasoning behind this is that the recommender agent, in particular the data in the playback buffer, needs a "restart" so that it can perform better in the first epochs of training when there is little data.

For training and testing, the input sample was split into training and testing samples in a ratio of 1:4. The following results were obtained on the test sample. We also took into account the cases when the generated recommendation is not in the dataset (one or more movies from the recommendation list are missing), in which case the result represents the residual for which the ratings exist.

Python 3.8 and the following versions of libraries were used:

- Keras '2.12.0';
- Tensorflow '2.12.0';
- Pandas '2.0.2';
- Numpy '1.12.5'.

A. Results for DDPG

The distribution of grades for movies recommended by the system based on the DDPG algorithm is shown in Fig. 3. Table II shows the results of the selected metrics. Figure 4 shows the last training iterations of the algorithm.

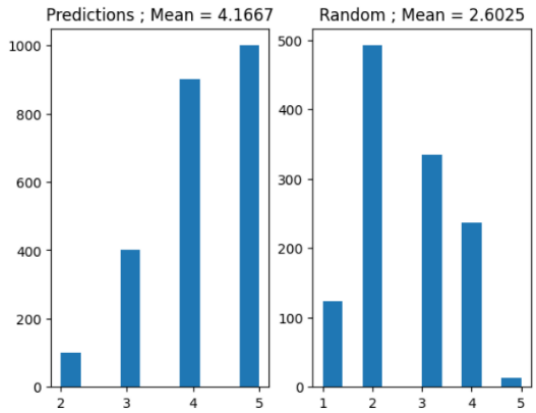

Fig. 3. Distribution of scores for users in the test sample for DDPG

TABLE II. RESULTS FOR DDPG

AVRR	HR	MRR	NDCG
4.1667	0.7916	0.5945	0.5357

```
Episode 82/100 Reward=1090 Q_value=913 Time=135s Loss=13.5732
Episode 83/100 Reward=1090 Q_value=912 Time=137s Loss=11.8960
Episode 84/100 Reward=1090 Q_value=912 Time=136s Loss=11.7201
Episode 85/100 Reward=1090 Q_value=918 Time=136s Loss=12.4271
Episode 86/100 Reward=1090 Q_value=920 Time=136s Loss=10.7970
Episode 87/100 Reward=1090 Q_value=919 Time=136s Loss=10.7166
Episode 88/100 Reward=1090 Q_value=912 Time=136s Loss=10.0800
Episode 89/100 Reward=1090 Q_value=915 Time=136s Loss=10.3653
Episode 90/100 Reward=1090 Q_value=906 Time=136s Loss=10.3260
Episode 91/100 Reward=1090 Q_value=907 Time=136s Loss=11.5718
Episode 92/100 Reward=1090 Q_value=918 Time=135s Loss=9.6398
Episode 93/100 Reward=1090 Q_value=912 Time=136s Loss=10.8653
Episode 94/100 Reward=1090 Q_value=902 Time=137s Loss=11.3878
Episode 95/100 Reward=1090 Q_value=910 Time=136s Loss=10.5785
Episode 96/100 Reward=1090 Q_value=911 Time=136s Loss=10.8166
Episode 97/100 Reward=1090 Q_value=903 Time=136s Loss=9.9071
Episode 98/100 Reward=1090 Q_value=912 Time=136s Loss=10.2206
Episode 99/100 Reward=1090 Q_value=912 Time=136s Loss=10.1447
Episode 100/100 Reward=1090 Q_value=912 Time=136s Loss=11.1764
```

Fig. 4. Latest iterations of DDPG model training

B. Results for TD3

The distribution of grades for movies recommended by the system based on the TD3 algorithm is shown in Fig. 5. Table III shows the results of the selected metrics. Figure 4 shows the last training iterations of the algorithm.

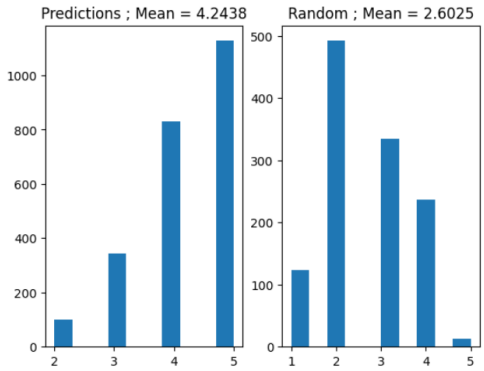

Fig. 5. Distribution of scores for users in the test sample for TD3

TABLE III. RESULTS FOR TD3

AVRR	HR	MRR	NDCG
4.2438	0.8212	0.6129	0.7357

```
Episode 82/100 Reward=1103 Q_value=1118 Time=136s Loss=6.6256
Episode 83/100 Reward=1102 Q_value=1124 Time=136s Loss=6.3834
Episode 84/100 Reward=1102 Q_value=1113 Time=136s Loss=6.1935
Episode 85/100 Reward=1102 Q_value=1117 Time=136s Loss=5.3433
Episode 86/100 Reward=1102 Q_value=1115 Time=136s Loss=5.3894
Episode 87/100 Reward=1102 Q_value=1116 Time=136s Loss=6.2048
Episode 88/100 Reward=1102 Q_value=1113 Time=136s Loss=6.3767
Episode 89/100 Reward=1102 Q_value=1115 Time=136s Loss=6.7249
Episode 90/100 Reward=1102 Q_value=1118 Time=136s Loss=5.1156
Episode 91/100 Reward=1102 Q_value=1113 Time=136s Loss=5.3250
Episode 92/100 Reward=1102 Q_value=1113 Time=136s Loss=6.4033
Episode 93/100 Reward=1102 Q_value=1112 Time=136s Loss=5.9472
Episode 94/100 Reward=1102 Q_value=1114 Time=136s Loss=4.7042
Episode 95/100 Reward=1102 Q_value=1115 Time=136s Loss=5.8889
Episode 96/100 Reward=1102 Q_value=1113 Time=136s Loss=5.1334
Episode 97/100 Reward=1102 Q_value=1105 Time=136s Loss=5.5885
Episode 98/100 Reward=1102 Q_value=1112 Time=136s Loss=5.4763
Episode 99/100 Reward=1102 Q_value=1109 Time=136s Loss=5.7275
Episode 100/100 Reward=1102 Q_value=1115 Time=136s Loss=5.6936
```

Fig. 6. Latest iterations of TD3 model training

VI. CONCLUSIONS

In the course of the research, an approach to recommendation systems based on reinforcement learning was developed. In its context, three specific modules were created: for coding states, for generating rewards, and for reproducing actions. These three components made it possible to move from the task of supervised learning to the task of RL, while taking advantage of both approaches within the same dataset.

The reward generation component allowed us to fully optimize the model to leave room for more powerful reinforcement learning algorithms, which showed quite good results.

A separate disadvantage is the rather long training time, which is especially noticeable in the TD3 model, although it showed better results.

REFERENCES

[1] J. A. Konstan and J. Riedl, "Recommender systems: from algorithms to user experience," *User Model User-Adapt Interact*, 22: 101–23. 2012. https://doi.org/10.1007/s11257-011-9112-x

[2] P. Pu, L. Chen, and R. Hu, "A user-centric evaluation framework for recommender systems," In: *Proceedings of the fifth ACM conference on Recommender Systems (RecSys'11), ACM*, New York, NY, USA; 2011, pp. 57–164. https://doi.org/10.1145/2043932.2043962

[3] R. Hu and P. Pu, "Potential acceptance issues of personality-ASED recommender systems," In: *Proceedings of ACM conference on recommender systems (RecSys'09),* New York City, NY, USA; October 2009. pp. 22–5. https://doi.org/10.1145/1639714.1639753

[4] B. Pathak, R.Garfinkel R. Gopal, R. Venkatesan, and F. Yin, "Empirical analysis of the impact of recommender systems on sales," *J Manage In form Syst*, 27(2): 159–88, 2010. https://doi.org/10.2753/MIS0742-1222270205

[5] Soufiene Jaffali, Salma Jamoussi, Kamel Smaili, and Abdelmajid Ben Hamadou, "Like-tasted user groups to predict ratings in recommender systems," *Social Netw. Analys*, Mining, vol. 10, no. 1, p. 42, 2020. https://doi.org/10.1007/s13278-020-00643-w

[6] K. Zhou, S.-H. Yang, and H. Zha, "Functional matrix factorizations for cold-start recommendation," in *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '11*, (New York, NY, USA), pp. 315–324, ACM, 2011. https://doi.org/10.1145/2009916.2009961

[7] X. Zhang, J. Cheng, T. Yuan, B. Niu, and H. Lu, "Semi-supervised discriminative preference elicitation for cold-start recommendation," in *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management, CIKM '13*, (New York, NY, USA), pp. 1813–1816, ACM, 2013. https://doi.org/10.1145/2505515.2507869

[8] P. Mazumdar, B. K. Patra, and K. S. Babu, "Cold-star point-of-interest recommendation through crowd sourcing," *ACM Trans. Web*, vol. 14, Aug. 2020. https://doi.org/10.1145/3407182

[9] N. K. Mishra, V. Mishra, and S. Chaturvedi, "Solving cold start problem using MBA," in *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*,

pp. 1598–1601, 2017. https://doi.org/10.1109/ICPCSI.2017.8391981

[10] N. Mishra, V. Mishra, and S. Chaturvedi, “Tools and techniques for solving cold start recommendation,” in *Proceedings of the 1st International Conference on Internet of Things and Machine Learning, IML ’17*, (New York, NY, USA), Association for Computing Machinery, 2017. https://doi.org/10.1145/3109761.3109772

[11] M. Saveskiand A. Mantrach, “Item cold-start recommendations: Learning local collective embeddings,” in *Proceedings of the 8th ACM Conference on Recommender Systems, RecSys ’14*, (New York, NY, USA), pp. 89–96, ACM, 2014. https://doi.org/10.1145/2645710.2645751

[12] Y. Rong, X. Wen, and H. Cheng, “A monte-carlo algorithm for cold start-recommendation,” in *Proceedings of the 23rd international conference on Worldwide web*, pp. 327–336, ACM, 2014. https://doi.org/10.1145/2566486.2567978

[13] A. da Costa, E. Fressato, F. Neto, M. Manzato, and R. Campello, “Case recommender: A flexible and extensible python frame work for recommender systems,” in *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys ’18*, (New York, NY, USA), p. 494–495, Association for Computing Machinery, 2018. https://doi.org/10.1145/3240323.3241611

[14] T. Kitazawaand M. Yui, “Query-based simple and scalable recommender systems with apache hive mall,” in *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys ’18*, (New York, NY, USA), p. 502–503, Association for Computing Machinery, 2018. https://doi.org/10.1145/3240323.3241592

[15] N. Mishra, S. Chaturvedi, V. Mishra, R. Srivastava, and P. Bargah, “Solving sparsity problem in rating-based movie recommendation system,” in *Computational Intelligence in Data Mining* (H. S. Beheraand D. P. Mohapatra, eds.), (Singapore), pp. 111–117, Springer Singapore, 2017. https://doi.org/10.1007/978-981-10-3874-7_11

[16] W. Pan, E. W. Xiang, N. N. Liu, and Q. Yang, "Transfer learning in collaborative filtering for sparsity reduction.," in *AAAI*, vol. 10, pp. 230–235, 2010. https://doi.org/10.1609/aaai.v24i1.7578.

[17] P. Adamopoulos and A. Tuzhilin, “On over-specialization and concentration bias of recommendations: Probabilistic neighborhood selection in collaborative filtering systems,” in *Proceedings of the 8th ACM Conference on Recommender systems*, pp. 153–160, ACM, 2014. https://doi.org/10.1145/2645710.2645752

[18] Z.-K. Zhang, C. Liu, Y.-C. Zhang, and T. Zhou, “Solving the cold-start problem in recommender systems with social tags,” *EPL (Euro physics Letters)*, vol. 92, no. 2, p. 28002, 2010. https://doi.org/10.1209/0295-5075/92/28002

[19] N. Lathia, S. Hailes, L. Capra, and X. Amatriain, “Temporal diversity in recommender systems,” in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pp. 210–217, ACM, 2010. https://doi.org/10.1145/1835449.1835486

[20] Lalita Sharma and Anju Gera. "A Survey of Recommendation System: Research Challenges," *International Journal of Engineering Trends and Technology (IJETT)*, vol. 4(5) 2013, pp.1989–1992.

[21] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Hybrid recommender systems: survey and experiments," *User Model User-adapted Interact*, 12 (4) (2002), pp. 331–370

[22] N. Friedman, D. Geiger, and M. Goldszmidt, ”Recommender systems survey Knowl-Based Syst,” *Bayesian net work classifiers*, vol.46, pp. 109–132, 2013, https://doi.org/10.1016/j.knosys.2013.03.012

[23] Mach Learn, 29 (2–3), 1997, pp. 131–163. https://doi.org/10.1023/A:1007465528199

[24] R. O. Duda, P. E. Hart, and D. G. Stork, Pattern classification John Wiley & Sons (2012).

[25] J. L. Herlocker, J. A. Konstan, L. G. Terveen, J. T. Riedl, “Evaluating collaborative filtering recommender systems ACM Trans *,” In form Syst*, 22 (1), 2004. https://doi.org/10.1145/963770.963772

[26] M. Montaner, B. Lopez, and J. L. Dela Rosa, “A Taxonomy of Recommender Agent son the Internet,” *Artificial Intelligence Review*, Kluwer Academic Publisher, 2003.

[27] Yan-Martin Tamm, Rinchin Damdinov, and Alexey Vasilev, Quality Metrics in Recommender Systems: Do We Calculate Metrics Consistently?

Received April 03, 2023

Sineglazov Victor. ORCID 0000-0002-3297-9060. Doctor of Engineering Science. Professor. Head of the Department of Aviation Computer-Integrated Complexes.
Faculty of Air Navigation Electronics and Telecommunications, National Aviation University, Kyiv, Ukraine.
Education: Kyiv Polytechnic Institute, Kyiv, Ukraine, (1973).
Research area: Air Navigation, Air Traffic Control, Identification of Complex Systems, Wind/Solar power plant, artificial intelligence.
Publications: more than 700 papers.
E-mail: svm@nau.edu.ua

Sheruda Andrii. Bachelor.
Department of Information Systems, Faculty of Informatics and Computer Science, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.
Education: National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine, (2022).
Research interests: artificial neural networks, artificial intelligence, distributed computing.
Publications: 1.
E-mail: sheruda.andrew@lll.kpi.ua

В. М. Синєглазов, А. В. Шеруда. Системи рекомендацій на основі посиленого навчання

Статтю присвячено проблемі побудови рекомендаційних систем на основі використання методів штучного інтелекту. У роботі проведено аналіз алгоритмів рекомендаційних систем, проаналізовано марківський процес прийняття рішень у контексті рекомендаційних систем. Розглянуто підходи до адаптації алгоритмів навчання з підкріпленням до завдання рекомендацій (перехід від задачі контрольованого навчання до завдання навчання з підкріпленням). Реалізовано алгоритми навчання з підкріпленням Deep Deterministic Policy Gradient та Twin Delayed DDPG із власним середовищем-імітацією реакції користувача та виконано порівняння результатів. Розроблено структуру рекомендаційної системи, у якій рекомендаційний агент генерує список пропозицій окремому користувачеві, використовуючи його попередню історію оцінок. У самій системі користувач має можливість взаємодії тільки з простором фільмів, що рекомендуються. Це можна порівняти з головною сторінкою YouTube, що є стрічкою з пропозиціями, у нас же користувач взаємодія тільки з цією стрічкою і його реакція на об'єкти в просторі рекомендацій потрапляє до рекомендаційного агента, який регулює параметри моделі в процесі навчання.

Ключові слова: машинне навчання; навчання з підкріпленням; системи рекомендацій; рекомендаційний агент; колаборативна фільтрація; Актор-Критик; явний зворотній зв’язок.

Синєглазов Віктор Михайлович. ORCID 0000-0002-3297-9060. Доктор технічних наук. Професор. Завідувач кафедри авіаційних комп’ютерно-інтегрованих комплексів.
Факультет аеронавігації, електроніки і телекомунікацій, Національний авіаційний університет, Київ, Україна.
Освіта: Київський політехнічний інститут, Київ, Україна, (1973).
Напрям наукової діяльності: аеронавігація, управління повітряним рухом, ідентифікація складних систем, вітроенергетичні установки, штучний інтелект.
Кількість публікацій: більше 700 наукових робіт.
E-mail: svm@nau.edu.ua

Шеруда Андрій Володимирович. Бакалавр.
Кафедра інформаційних систем, Факультет інформатики та обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.
Освіта: Національний технічний університет Україні «Київський політехнічний інститут імені Ігоря Сікорського», (2022).
Напрям наукової діяльності: штучні нейронні мережі, штучний інтелект, розподіленні обчислення.
Кількість публікацій: 1.
E-mail: sheruda.andrew@lll.kpi.ua