Recommender systems (RSs) have become an inseparable part of our everyday lives. They help us find our favorite items to purchase, our friends on social networks, and our favorite movies to watch.
The massive volume of information available on the web leads to the problem of *information overload*, which makes it difficult for a decision maker to make right decisions. The realization of this in our everyday lives is when we face a long list of items in an online shopping store; the more items in the list, the tougher it becomes to select among them. Recommender systems (RSs) are software tools and algorithms that have been developed with the idea of helping users find their items of interest, through predicting their preferences or ratings on items. In fact, the idea is to know the users to some extent, i.e., making a *user profile* based on their feedback on items, and to recommending those items that match their profile. Today, RSs are an essential part of most giant companies, like Google, Facebook, Amazon, and Netflix, and employed in a wide range of applications, including entertainment, e-commerce, news, e-learning, and healthcare.
*Recommendation system are widely used in e-commerce that is a part of ebusiness. It helps users locate information or products that they would like to make offers.*The growth of information on World Wide Web make users more difficult to search for relevant information as the amount of product in e-business increases rapidly. Customers suffer from searching for interested products. To avoid this problem, many websites use recommendation system to help customer finding the satisfy products. Recommendation systems are categorized into two major classes: content-based filtering and collaborative filtering [1]. In content-based filetering, the system tries to match the content of product with user profile, both content of product and user profile represented by keywords. Robin van Meteren and Maarten van Someren proposed PRES that use content-based filtering techniques to suggest document that relevance to user profile. The user profile was created by user feedback. In collaborative filtering, the system try to match user pattern with another users that had the same taste then predict the most user's interest to items. GroupLens [2] is a collaborative filtering of netnews, by rating articles after read, to suggest customers the articles that related with their interests.

Traditionally, the recommendation problem was considered to be a classification or prediction problem, but it is now widely agreed that formulating it as a sequential decision problem can better reflect the user-system interaction. Therefore, it can be formulated as a Markov decision process (MDP) and be solved by reinforcement learning (RL) algorithms. Unlike traditional recommendation methods, including collaborative filtering and content-based filtering, RL is able to handle the sequential, dynamic user-system interaction and to take into account the long-term user engagement. Although the idea of using RL for recommendation is not new and has been around for about two decades, it was not very practical, mainly because of scalability problems of traditional RL algorithms. However, a new trend has emerged in the field since the introduction of deep reinforcement learning (DRL), which made it possible to apply RL to the recommendation problem with large state and action spaces.

Reinforcement Learning

Reinforcement learning is one of powerful machine learning algorithm. Learning from reinforcement is a trial-anderror learning scheme.

Reinforcement learning (RL) is a machine learning field that studies problems and their solutions in which agents, through interaction with their environment, learn to maximize a numerical reward. According to Sutton and Barto, three characteristics distinguish an RL problem: (1) the problem is closed-loop, (2) the learner does not have a tutor to teach it what to do, but it should figure out what to do through trial-and-error, and (3) actions influence not only the short-term results, but also the long-term ones. The most common interface to model an RL problem is the *agent environment* interface. The learner or decision maker is called *agent* and the *environment* is
everything outside the agent. Accordingly, at time step , the agent sees some representations/information about the environment, called *state*, and based on the current state it takes an *action*. On taking this action, it receives a numerical *reward* from the environment and finds itself in a new state.
More formally, the RL problem is typically formulated as a Markov decision process (MDP) in the form of a tuple (*S, A, R, P,*), where *S* is the set of all possible states, *A* is the set of available actions in all states, *R* is the reward
function, *P* is the transition probability, and is the discount factor.
The main elements of an RL system are:

- **Policy***:* policy is usually indicated by and gives the probability of taking action when the agent is in state . Regarding the policy, RL algorithms can be generally divided into *on-policy* and *off-policy* methods. In the former, RL methods aim at evaluating or improving the policy they are using to make decisions. In the latter, they improve or evaluate a policy that is different from the one used to generate the data.
- **Reward signal***:* upon selecting actions, the environment provides a numerical reward to inform the agent how good or bad are the actions selected.
- **Value function***:* the reward signal is merely able to tell what is good immediately, but the value function defines what is good in the long run.
- **Model***:* model provides the opportunity to make inferences about the behavior of the environment. For instance,
the model can predict next state and next reward in a given state and action.
- **State** *S***:** a state $\in$ *S* is defined as the user preferences and their past history with the system.
- **Action** *A***:** an action $\in$ *A* is to recommend an item to the user at time step .
- **Reward** *R***:** the RL agent receives reward r(s_t, a_t) $\in$ *R* based on the user feedback on the recommendation
provided. Agent can learn to perform an appropiate action by recieving evaluation feedback. The objective is trying to maximize the expected sum of future values for each state. One major component of reinforcement algorithm is On-Policy TD control, called SARSA method. It consists of state-action pair and we can learn from the changing of value state Q(s,a) between stateaction pair to another state-action pair. The value state defined as:

$$Q(s_t, a_t) \longleftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad \text{......................1}$$

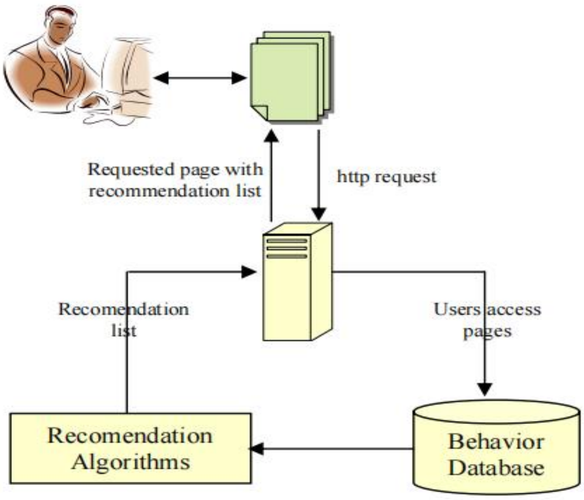
where st is the state of agent at time t, at is the action of agent at time t, rt+1 is reward of state s that action a, α is learning rate (0≤α<1) and γ is discount rate (0≤γ<1).

The Sarsa control algorithm is given as :

Initialize Q(s, a)
Repeat
Initial s
Choose a from s using policy
Repeat
Take action a, observe r′, s′
Choose a′ from s′ using policy
Update Q(s, a)
s′←s; a′←a;
until s is terminal

First, we initialize value of Q(s,a) and choose the initial state s. Second, we select action *a* at state *s* using policy. The policy can be greedy or ε-greedy policy. Next, repeat take action *a*, observe the reward *r* and the next state *s'*, choose *a'* from *s'* using policy for compute the value of future next state and then update the Q(s,a) of current state and change *s*←*s'*, *a*←*a'*.

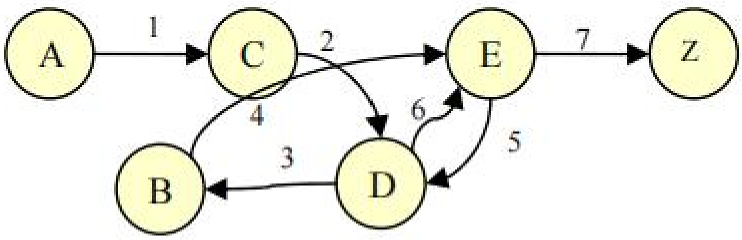
System Architecture



1. Behavior Database. The system keeps behavior of customers into two groups. First, the global behavior collected behavior of all customers, we can know another customers direction. Second, the local behavior collected behavior of each customer, we can know which product bought by them.

2. Recomendation Algorithms. It’s a part for learning by reinforcement learning and send recommendation list to users. When the customers login to system and open a page. The pages can be viewed as states s of the system and links in a page can be viewed as action a of state. The system puts the state diagram (page-to-page) as shown in the figure below and the changing value between states into Q-matrix.

An example of a state diagram of user behavior.



The system has two type of Q-matrix. First, global Q-matrix keeps the changeablility of whole system. The global Q-matrix can tell you the trend of customers or most popular products. Second, local Qmatrix keeps the changeablility of each customer. The local Q-matrix likes customer profile that record customer browsing behavior. To update of both Q-matrix system will get reward from click on the products and make offered, we call customer feedbacks. Customer feedbacks are important part of recommendation system. The Customer feedbacks may be explicit and implicit. Customers can send the explicit feedback by rating the products. The system will update Q-Matrix of that produce page. For example, the customer may give rate 3 out of 5. Although explicit rating is accurate, there are few customers who gave rating for produces after they used. The system needs another feedback from customers. Implicit feedback is percieved by keeping customer's behavior. It has two type of implicit feedback. First, when customer changes state that means click on product page. Second, when customer changes state to final state that means product added into shopping card and bought its.
In Table.1 shown you a local Q-matrix of customer who has the changing state like in Figure 3. The customer logins into website and click on product A. Then he/she has sequence of the changing state like A→C, C→D, D→B, B→E, E→D, D→E and bought product E. When customer click on each produce page, system will update both Q-matrix by plus 1 and when customer bought the product plus 3 to that product.

Table 1. Q-matrix of user

Action / State	A	B	C	D	E
A			1		
B					1
C				1	
D		1			1+1+3
E				1	

To predict next state or next product that customers may prefer to offer, the system will rank products. The ranking system separated into 2 parts. First

part is the ranking of whole system, global ranking, system uses data from global Q-matrix to choose the next state by using ε-greedy policy.
If you maintain estimates of the action values, then at any time there is at least one action whose estimated value is greatest. We call this a greedy action. If you select a greedy action, we say that you are exploiting your current knowledge of the values of the actions. If instead you select one of the nongreedy actions, then we say you are exploring because this enables you to improve your estimate of the nongreedy action's value. The ε is a small probability to select an action at random. The advantage of ε-greedy policy over greedy policy are the ε-greedy policy continue to explore and improve their chances of recognizing the product that customer may make offered. The greedy policy will choose only the state that has maximum values. If the ε=0.2, the method explores more than, and usually finds the optimal action earlier, but never selects it more than 81% of the time. The ε=0.1 method improves more slowly. For example if the system have 5 ranks and ε=0.1. In each position of the system will have chance to choose the highest Q-value equal to 90% and choose another equal to 10%. To do like this the system gives a chance for new products or product that have few clicks on but may be match with your interest.
Second part is the ranking of each customer, local ranking, the system considered from local Q-matrix. To choose the next state, the system uses inverse ε- greedy policy. The states that customer ever visit, system will decrease important and gives a chance to explore other products.
After that system finds Qtotal by using eq 2.

$$Q_{total}=Q_{local}+wQ_{global} \quad \text{.........} 2$$

where w is the weight of Qglobal and $w \in (0,1]$.
Final, system will rank produces by using Qtotal and recoment to customers.

Experiments

The objective of experiment was to find relationship between ε and user click rate, the click rate is a measure of how many of the presented products in recomendation list that customer clicks on. Tab. 2 shows the average customer clicks rate where w=0.8.

Table2. The average customer click rate.

degree of ε	average custumer click rate
0.10	62.75 %
0.15	68.50 %
0.20	75.50 %
0.25	71.30 %
0.30	60.75 %
0.40	58.00 %
0.50	52.50 %

We found that ε = 0.2 can preserve balancing exploration and exploitation power. If ε less than 0.2 the system will exploit to the trend of system and gave a small chance to explore new product. So in the early time of user login the clicks rate was high and after that the system showed that the clicks rate dropped for the same previous product. Although Qlocal try to promote the new product, it is less effective than Qglobal. If ε more than 0.2 the system will too much explore the products. The system may include the product that does not match the customer.

The current challenges or ethical issues.
RL Challenges. There are some possible challenges when applying RL to any problem. A challenge well-known as Deadly Triad states that there is a hazard of instability and divergence when combining three elements in RL: function approximation, bootstrapping, and off-policy training.
Another challenge in RL is sample inefficiency, specifically in model-free RL algorithms. Current model-free RL algorithms need a considerable amount of agent-environment interaction in order to learn useful states. Moreover, since deep reinforcement learning is based on deep learning, it consequently inherits the famous feature of neural networks, i.e., being black-box. It is not obvious how weights and activations are changed, which makes them uninterpretable. The classical problem of exploration vs exploitation is still a challenge in RL and effective exploration is an open research problem
One of the challenges is the need to balance exploration and exploitation in the recommendation system. Finally, the problem of reward formulation in RL is a challenge and designing a good reward function is not very clear or straightforward.

Conclusions and Future Work .

In this paper a deep insight was derived for a general framework for recommendation system based on reinforcement learning. The system can learn direct from customer's behavior. The learning process is using SARSA method and ε-greedy policy. The system consists of two parts, global model and local model. One important aspect of learning method is balance of exploration and exploitation. The ε-greedy policy can give a chance to explore new produces, but the same time its exploit to the trend of system. We showed this in a simple experiment about the effect of ε value and user click rate. The result of this experiment can explain that if you explore too much it has a chance to comment the uninterested product to users, if you exploit too much it has a chance to stick with other opinion and never see the other products. In the real world, it requires more space to keep global state and local state of all users. We plan to continue reducing the space by using another data structure. We also study in the effect of w to find the optimal w for Qtotal.

To further improve the current methods and techniques, it is essential to address the challenges of maintaining global and local state information for all users. This could involve optimizing the weight for the Qtotal calculation and exploring data structure improvements to reduce space requirements. Additionally, further research could focus on finding the optimal ε value for balancing exploration and exploitation in the recommendation system.

In conclusion, this paper presents a comprehensive overview of the recommendation system using reinforcement learning, its architecture, and the results of experiments, providing valuable insights for the development of advanced recommendation systems in e-commerce and e-business. The application of reinforcement learning in this context offers a promising approach to personalized recommendations and user satisfaction. However, addressing the challenges and continuing to improve the methods and techniques will be crucial for the future development of recommendation systems.

[Optional challenge] .

comparisons of reinforcement learning techniques.

I investigated different recommendation system model method using python programming. I attached the python file to my report submission

1) Random Model method on Recommendation System.

This algorithm predicts a random rating based on the distribution of the training set, which is assumed to be normal. This is one of the most basic algorithms.

Below is the evaluation score of the model .

```
Evaluating RMSE of algorithm NormalPredictor on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.5197  1.5194  1.5215  1.4993  1.5251  1.5170  0.0091
Fit time          0.15    0.18    0.18    0.18    0.17    0.17    0.01
Test time         0.28    0.16    0.15    0.25    0.15    0.20    0.06
```

2) User-Based Collaborative Filtering (UB-CF) Model method on Recommendation System.

This algorithm takes a particular user, find users who have similar ratings, and then recommend items that those similar users liked.

To implement a user based collaborative filtering we will use the Nearest Neighbor algorithm. This algorithm needs three tasks:

a) Find the K-nearest neighbors (KNN), that are similar to the user A, using a similarity function to measure the distance between each pair of users.
b) Predict the rating that user A will give to all items the K neighbors have consumed but A has not.
c) Select top-n rated movies.

Below is the evaluation score of the model .

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.9783	0.9825	0.9774	0.9837	0.9694	0.9782	0.0050
Fit time	0.52	0.52	0.51	0.48	0.50	0.51	0.01
Test time	4.31	4.36	4.50	4.34	4.43	4.39	0.07

3) User-Based Collaborative Filtering (UB-CF) Model method on Recommendation System.

In this approach, instead of focus on similar users to our user A, we will focus on what items from all the options, are more similar to what we know the user A enjoys.

The algorith will work in three tasks:

a) Calculate the similarity between any two items and fill up the item-item similarity matrix.
b) Predict the ratings of movies that are rated by user A.
c) Select top-n rated movies for A.

Below is the evaluation score of the model .

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.9756	0.9749	0.9718	0.9736	0.9751	0.9742	0.0013
Fit time	0.79	0.78	0.73	0.71	0.73	0.75	0.03
Test time	4.86	4.92	4.95	4.89	5.10	4.94	0.08

4)Matrix Factorization Model method on Recommendation System..

In the matrix factorization model the recommendations are based on the discovery of latent features of the interactions between users and items.

Below is the evaluation score of the model .

```
Evaluating RMSE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9316  0.9390  0.9371  0.9362  0.9399  0.9368  0.0029
Fit time          6.04    6.04    6.07    6.03    6.04    6.04    0.01
Test time         0.18    0.17    0.31    0.17    0.17    0.20    0.06
```

Comparison for each models:

```
[ ] means = [round(model_random_results['test_rmse'].mean(),4),round(model_user_based_results['test_rmse'].mean(),4), round(model_item_based_results['test_rmse'].mean(),4),
    table = pd.Series(means, ['Random','User-based', 'Item-based', 'Matrix factorization'])
    print("\t RMSE Means for each model\n")
    print(table)

         RMSE Means for each model

    Random                  1.5170
    User-based              0.9782
    Item-based              0.9742
    Matrix factorization    0.9368
    dtype: float64
```

The report contains the .ipynb file for the full analysis of the model.

References.

[1] Cisco Visual Networking Index. The zettabyte era–trends and analysis. *Cisco white paper*, 2013.

[2] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[3] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. 2011.

[4] George Lekakos and Petros Caravelas. A hybrid approach for movie recommendation. *Multimedia tools and applications*, 36(1):55–70, 2008.

[5] Hung-Chen Chen and Arbee LP Chen. A music recommendation system based on music data grouping and user interests. In *CIKM'01*, pages 231–238, 2001.

[6] Xuan Zhu, Yuan-Yuan Shi, Hyoung-Gook Kim, and Ki-Wan Eom. An integrated music recommendation system. *IEEE Transactions on Consumer Electronics*, 52(3):917–925, 2006.

[7] J Ben Schafer, Joseph Konstan, and John Riedl. Recommender systems in e-commerce. In *EC'99*, pages 158–166, 1999.

[8] Mozhgan Karimi, Dietmar Jannach, and Michael Jugovac. News recommender systems–survey and roads ahead. *Information Processing & Management*, 54(6):1203–1227, 2018.

[9] Aleksandra Klašnja-Milićević, Mirjana Ivanović, and Alexandros Nanopoulos. Recommender systems in e-learning environments: a survey of the state-of-the-art and possible extensions. *Artificial Intelligence Review*, 44(4):571–604, 2015.
[10] Emre Sezgin and Sevgi Özkan. A systematic literature review on health recommender systems. In *EHB'13*, pages 1–4, 2013.
[11] Netflix update: Try this at home. https://sifter.org/~simon/journal/20061211.html.
[12] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-based systems*, 46:109–132, 2013.
[13] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. MIT press Cambridge, 2016.
[14] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.
[15] Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.
[16] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *AAAI'18*, 2018.
[17] Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017(19):70–76, 2017.
[18] Changxi You, Jianbo Lu, Dimitar Filev, and Panagiotis Tsiotras. Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning. *Robotics and Autonomous Systems*, 114:1–18, 2019.
[19] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*, 32(11):1238–1274, 2013.
[20] Richard Meyes, Hasan Tercan, Simon Roggendorf, Thomas Thiele, Christian Büscher, Markus Obdenbusch, Christian Brecher, Sabina Jeschke, and Tobias Meisen. Motion planning for industrial robots using reinforcement learning. *CIRP'17*, 63:107–112, 2017.
[21] Zhengyao Jiang, Dixing Xu, and Jinjun Liang. A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059*, 2017.
[22] Arthur Guez, Robert D Vincent, Massimo Avoli, and Joelle Pineau. Adaptive treatment of epilepsy via batch-mode reinforcement learning. In *AAAI'08*, pages 1671–1678, 2008.
[23] Omer Gottesman, Fredrik Johansson, Matthieu Komorowski, Aldo Faisal, David Sontag, Finale Doshi-Velez, and Leo Anthony Celi. Guidelines for reinforcement learning in healthcare. *Nature Medicine*, 25(1):16–18, 2019.
[24] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.
[25] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. Top-k off-policy correction for a reinforce recommender system. In *WSDM'19*, pages 456–464, 2019.

[26] Aleksandrs Slivkins. Introduction to multi-armed bandits. *arXiv preprint arXiv:1904.07272*, 2019.
[27] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
[28] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW'10*, pages 661–670, 2010.
[29] Shuai Li, Alexandros Karatzoglou, and Claudio Gentile. Collaborative filtering bandits. In *SIGIR'16*, pages 539–548, 2016.
[30] Charu C Aggarwal. *Recommender systems*. Springer, 2016.
[31] M. Mehdi Afsar, Trafford Crump, and Behrouz Far. An exploration on-demand article recommender system for cancer patients information provisioning. In *FLAIRS'21*, volume 34, 2021.

[32] Gangan Elena, Kudus Milos, and Ilyushin Eugene. Survey of multiarmed bandit algorithms applied to recommendation systems. *International Journal of Open Information Technologies*, 9(4):12–27, 2021.
[33] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009.
[34] Yue Shi, Martha Larson, and Alan Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)*, 47(1):1–45, 2014.
[35] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.
[36] Feng Xia, Nana Yaw Asabere, Ahmedin Mohammed Ahmed, Jing Li, and Xiangjie Kong. Mobile multimedia recommendation in smart communities: A survey. *IEEE Access*, 1:606–624, 2013.
[37] Yashar Deldjoo, Markus Schedl, Paolo Cremonesi, and Gabriella Pasi. Recommender systems leveraging multimedia content. *ACM Computing Surveys (CSUR)*, 53(5):1–38, 2020.
[38] Yongfeng Zhang and Xu Chen. Explainable recommendation: A survey and new perspectives. *arXiv preprint arXiv:1804.11192*, 2018.
[39] Joeran Beel, Bela Gipp, Stefan Langer, and Corinna Breitinger. paper recommender systems: A literature survey. *International Journal on Digital Libraries*, 17(4):305–338, 2016.
[40] Xiangyu Zhao, Long Xia, Jiliang Tang, and Dawei Yin. Deep reinforcement learning for search, recommendation, and online advertising: a survey. *ACM SIGWEB Newsletter*, pages 1–15, 2019.
[41] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. Sequence-aware recommender systems. *ACM Computing Surveys (CSUR)*, 51(4):1–36, 2018.
[42] Shoujin Wang, Longbing Cao, Yan Wang, Quan Z Sheng, Mehmet A Orgun, and Defu Lian. A survey on session-based recommender systems. *ACM Computing Surveys (CSUR)*, 54(7):1–38, 2021.
[43] Shi-Yong Chen, Yang Yu, Qing Da, Jun Tan, Hai-Kuan Huang, and Hai-Hong Tang. Stabilizing reinforcement learning in dynamic environment with application to online recommendation. In *SIGKDD'18*, pages 1187–1196, 2018.
[44] Sungwoon Choi, Heonseok Ha, Uiwon Hwang, Chanju Kim, Jung-Woo Ha, and Sungroh Yoon. Reinforcement learning based recommender system

using biclustering technique. *arXiv preprint arXiv:1801.05532*, 2018.
[45] Isshu Munemasa, Yuta Tomomatsu, Kunioki Hayashi, and Tomohiro Takagi. Deep reinforcement learning for recommender systems. In *ICOIACT'18*, pages 226–233, 2018.
[46] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. Deep reinforcement learning for page-wise recommendations. In *RecSys'18*, pages 95–103, 2018.
[47] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. Drn: A deep reinforcement learning framework for news recommendation. In *WWW'18*, pages 167–176, 2018.
[48] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. Recommendations with negative feedback via pairwise deep reinforcement learning. In *SIGKDD'18*, pages 1040–1048, 2018.
[49] Omar Moling, Linas Baltrunas, and Francesco Ricci. Optimal radio channel recommendations with explicit and implicit feedback. In *RecSys*, pages 75–82, 2012.
[50] Guy Shani, David Heckerman, and Ronen I Brafman. An mdp-based recommender system. *Journal of Machine Learning Research*, 6:1265–1295, 2005.
[51] Binbin Hu, Chuan Shi, and Jian Liu. Playlist recommendation based on reinforcement learning. In *ICIS'17*, pages 172–182, 2017.
[52] Xiangyu Zhao, Liang Zhang, Long Xia, Zhuoye Ding, Dawei Yin, and Jiliang Tang. Deep reinforcement learning for list-wise recommendations. *arXiv preprint arXiv:1801.00209*, 2017.
[53] Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.
[54] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.
[55] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.
[56] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. 2011.
[57] Christopher Watkins. *Learning from delayed rewards*. University of Cambridge, 1989.
[58] Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, 1994.
[59] Anton Schwartz. A reinforcement learning method for maximizing undiscounted rewards. In *ICML'93*, pages 298–305, 1993.
[60] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.
[61] Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. Monte-carlo tree search: A new framework for game ai. *AIIDE*, 8:216–217, 2008.
[62] David Silver et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[63] Geoffrey J Gordon. *Approximate solutions to Markov decision processes*. Carnegie Mellon University, 1999.
[64] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

[65] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.
[66] Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.
[67] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.
[68] Volodymyr Mnih et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
[69] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.
[70] Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Connectionist Models Summer School Hillsdale*, 1993.
[71] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *arXiv preprint arXiv:1509.06461*, 2015.
[72] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *ICML'16*, pages 1995–2003, 2016.
[73] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.
[74] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
[75] David Silver et al. Deterministic policy gradient algorithms. In *ICML'14*, 2014.
[76] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
[77] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML'15*, pages 1889–1897, 2015.
[78] Mohammad Mehdi Afsar, Trafford Crump, and Behrouz Far. Sample Efficiency in Deep Reinforcement Learning based Recommender Systems with Imitation Learning. In *Canadian Conference on Artificial Intelligence*, 2022.
[79] A. Zimdars, D. M. Chickering, and C. Meek. Using temporal data for making recommendations. In *UAI'01*, pages 580–588, 2001.
[80] Eugene Ie et al. Reinforcement learning for slate-based recommender systems: A tractable decomposition and practical methodology. *arXiv preprint arXiv:1905.12767*, 2019.
[81] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence

modeling. *arXiv preprint arXiv:1412.3555*, 2014.
[82] Joeran Beel and Stefan Langer. A comparison of offline evaluations, online evaluations, and user studies in the context of research-paper recommender systems. In *TPDL'15*, pages 153–168, 2015.
[83] Thorsten Joachims, Dayne Freitag, Tom Mitchell, et al. Webwatcher: A tour guide for the world wide web. In *IJCAI'97*, pages 770–777, 1997.
[84] Mircea Preda and Dan Popescu. Personalized web recommendations: supporting epistemic information about end-users. In *WI'05*, pages 692–695, 2005.
[85] Nima Taghipour, Ahmad Kardan, and Saeed Shiry Ghidary. Usage-based web recommendations: a reinforcement learning approach. In *RecSys'07*, pages 113–120, 2007.
[86] Tariq Mahmood and Francesco Ricci. Learning and adaptivity in interactive recommender systems. In *EC'07*, pages 75–84, 2007.
[87] Nima Taghipour and Ahmad Kardan. A hybrid web recommender system based on q-learning. In *SAC'08*, pages 1164–1168, 2008.
[88] Tariq Mahmood and Francesco Ricci. Improving recommender systems with adaptive conversational strategies. In *HT'09*, pages 73–82, 2009.
[89] Chung-Yi Chi, Richard Tzong-Han Tsai, Jeng-You Lai, and Jane Yung-jen Hsu. A reinforcement learning approach to emotion-based automatic playlist generation. In *TAAI'10*, pages 60–65, 2010.
[90] Susan M Shortreed, Eric Laber, Daniel J Lizotte, T Scott Stroup, Joelle Pineau, and Susan A Murphy. Informing sequential clinical decision-making through reinforcement learning: an empirical study. *Machine learning*, 84(1-2):109–136, 2011.
[91] Yufan Zhao, Donglin Zeng, Mark A Socinski, and Michael R Kosorok. Reinforcement learning strategies for clinical trials in nonsmall cell lung cancer. *Biometrics*, 67(4):1422–1433, 2011.
[92] Tariq Mahmood, Ghulam Mujtaba, and Adriano Venturini. Dynamic personalization in conversational recommender systems. *Information Systems and e-Business Management*, 12(2):213–238, 2014.
[93] Elad Liebman, Maytal Saar-Tsechansky, and Peter Stone. Dj-mc: A reinforcement-learning agent for music playlist recommendation. *arXiv preprint arXiv:1401.1880*, 2014.
[94] Georgios Theocharous, Philip S Thomas, and Mohammad Ghavamzadeh. Personalized ad recommendation systems for life-time value optimization with guarantees. In *IJCAI'15*, 2015.
[95] Zhongqi Lu and Qiang Yang. Partially observable markov decision process for recommender systems. *arXiv preprint arXiv:1608.07793*, 2016.
[96] Yang Zhang, Chenwei Zhang, and Xiaozhong Liu. Dynamic scholarly collaborator recommendation via competitive multi-agent reinforcement learning. In *RecSys'17*, pages 331–335, 2017.
[97] Wacharawan Intayoad, Chayapol Kamyod, and Punnarumol Temdee. Reinforcement learning for online learning recommendation system. In *GWS*, pages 167–170, 2018.

[98] Jia-Wei Chang, Ching-Yi Chiou, Jia-Yi Liao, Ying-Kai Hung, Chien-Che Huang, Kuan-Cheng Lin, and Ying-Hung Pu. Music recommender using

deep embedding-based features and behavior-based reinforcement learning. *Multimedia Tools and Applications*, pages 1–28, 2019.
[99] Jing Chen and Wenjun Jiang. Context-aware personalized poi sequence recommendation. In *iSCI'19*, pages 197–210. Springer, 2019.
[100] Yu Wang. A hybrid recommendation for music based on reinforcement learning. In *PAKDD'20*, pages 91–103, 2020.
[101] Marios Kokkodis and Panagiotis G Ipeirotis. Demand-aware career path recommendations: A reinforcement learning approach. *Management Science*, 67(7):4362–4383, 2021.
[102] Bamshad Mobasher, Robert Cooley, and Jaideep Srivastava. Automatic personalization based on web usage mining. *Communications of the ACM*, 43(8):142–151, 2000.
[103] Andriy Mnih and Russ R Salakhutdinov. Probabilistic matrix factorization. In *NIPS'08*, pages 1257–1264, 2008.
[104] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS'13*, pages 3111–3119, 2013.
[105] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.
[106] Hui Li, Tsz Nam Chan, Man Lung Yiu, and Nikos Mamoulis. Fexipro: fast and exact inner product retrieval in recommender systems. In *SIGMOD'17*, pages 835–850, 2017.
[107] Richard E Bellman. *Adaptive control processes*. Princeton university press, 2015.
[108] Andrew G Barto. Reinforcement learning and dynamic programming. In *Analysis, Design and Evaluation of Man-Machine Systems*, pages 407–412. 1995.
[109] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *ECML'06*, pages 282–293, 2006.
[110] Movielens. https://grouplens.org/datasets/movielens/.
[111] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *ISMIR'11*, pages 591–596, 2011.
[112] Peter Sunehag, Richard Evans, Gabriel Dulac-Arnold, Yori Zwols, Daniel Visentin, and Ben Coppin. Deep reinforcement learning with attention for slate markov decision processes with high-dimensional states and actions. *arXiv preprint arXiv:1512.01124*, 2015.
[113] Shamim Nemati, Mohammad M Ghassemi, and Gari D Clifford. Optimal medication dosing from suboptimal clinical examples: A deep reinforcement learning approach. In *EMBC'16*, pages 2978–2981, 2016.
[114] Claudio Greco, Alessandro Suglia, Pierpaolo Basile, and Giovanni Semeraro. Converse-et-impera: Exploiting deep learning and hierarchical reinforcement learning for conversational recommender systems. In *AIxIA'17*, pages 372–386, 2017.
[115] Aniruddh Raghu, Matthieu Komorowski, Imran Ahmed, Leo Celi, Peter Szolovits, and Marzyeh Ghassemi. Deep reinforcement learning for sepsis treatment. *arXiv preprint arXiv:1711.09602*, 2017.
[116] Lu Wang, Wei Zhang, Xiaofeng He, and Hongyuan Zha. Supervised reinforcement learning with recurrent neural network for dynamic treatment

recommendation. In *SIGKDD'18*, pages 2447–2456, 2018.
[117] Yueming Sun and Yi Zhang. Conversational recommender system. In *SIGIR'18*, pages 235–244, 2018.
[118] Chenfei Zhao and Lan Hu. Capdrl: A deep capsule reinforcement learning for movie recommendation. In *PRICAI'19*, pages 734–739. Springer, 2019.
[119] Lixin Zou, Long Xia, Zhuoye Ding, Jiaxing Song, Weidong Liu, and Dawei Yin. Reinforcement learning to optimize long-term user engagement in recommender systems. In *SIGKDD'19*, pages 2810–2818, 2019.
[120] Rong Gao, Haifeng Xia, Jing Li, Donghua Liu, Shuai Chen, and Gang Chun. Drcgr: Deep reinforcement learning framework incorporating cnn and gan-based for interactive recommendation. In *ICDM'19*, pages 1048–1053, 2019.
[121] Tong Yu, Yilin Shen, Ruiyi Zhang, Xiangyu Zeng, and Hongxia Jin. Vision-language recommendation via attribute augmented multimodal reinforcement learning. In *MM'19*, pages 39–47, 2019.
[122] Daisuke Tsumita and Tomohiro Takagi. Dialogue based recommender system that flexibly mixes utterances and recommendations. In *WI'19*, pages 51–58, 2019.
[123] Jing Zhang, Bowen Hao, Bo Chen, Cuiping Li, Hong Chen, and Jimeng Sun. Hierarchical reinforcement learning for course recommendation in moocs. In *AAAI'19*, pages 435–442, 2019.
[124] Jason Zhang, Junming Yin, Dongwon Lee, and Linhong Zhu. Deep reinforcement learning for personalized search story recommendation. *Journal of Environmental Sciences*, 2019.
[125] Dong Liu and Chenyang Yang. A deep reinforcement learning approach to proactive content pushing and recommendation for mobile users. *IEEE Access*, 7:83120–83136, 2019.
[126] Zhang Yuyan, Su Xiayao, and Liu Yong. A novel movie recommendation system based on deep reinforcement learning with prioritized experience replay. In *ICCT'19*, pages 1496–1500, 2019.
[127] Tao Gui, Peng Liu, Qi Zhang, Liang Zhu, Minlong Peng, Yunhua Zhou, and Xuanjing Huang. Mention recommendation in twitter with cooperative multi-agent reinforcement learning. In *SIGIR'19*, pages 535–544, 2019.
[128] Ruiyi Zhang, Tong Yu, Yilin Shen, Hongxia Jin, and Changyou Chen. Text-based interactive recommendation via constraint-augmented reinforce ment learning. In *NIPS'19*, 2019.
[129] Yu Lei and Wenjie Li. Interactive recommendation with user-specific deep reinforcement learning. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(6):1–15, 2019.

[130] Xueying Bai, Jian Guan, and Hongning Wang. Model-based reinforcement learning with adversarial training for online recommendation. *arXiv preprint arXiv:1911.03845*, 2019.
[131] Floris Den Hengst, Mark Hoogendoorn, Frank Van Harmelen, and Joost Bosman. Reinforcement learning for personalized dialogue management. In *WI'19*, pages 59–67, 2019.
[132] Yikun Xian, Zuohui Fu, S Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. Reinforcement knowledge graph reasoning for explainable recommendation. In *SIGIR'19*, pages 285–294, 2019.

[133] Haokun Chen, Xinyi Dai, Han Cai, Weinan Zhang, Xuejian Wang, Ruiming Tang, Yuzhou Zhang, and Yong Yu. Large-scale interactive recommen dation with tree-structured policy gradient. In *AAAI'19*, volume 33, pages 3312–3320, 2019.
[134] Lixin Zou, Long Xia, Zhuoye Ding, Dawei Yin, Jiaxing Song, and Weidong Liu. Reinforcement learning to diversify top-n recommendation. In *DASFAA'19*, pages 104–120, 2019.
[135] Xinshi Chen, Shuang Li, Hui Li, Shaohua Jiang, Yuan Qi, and Le Song. Generative adversarial user model for reinforcement learning based recommendation system. In *ICML'19*, pages 1052–1061, 2019.
[136] Weiping Song, Zhijian Duan, Ziqing Yang, Hao Zhu, Ming Zhang, and Jian Tang. Explainable knowledge graph-based recommendation via deep reinforcement learning. *arXiv preprint arXiv:1906.09506*, 2019.
[137] Lixin Zou, Long Xia, Pan Du, Zhuo Zhang, Ting Bai, Weidong Liu, Jian-Yun Nie, and Dawei Yin. Pseudo dyna-q: A reinforcement learning framework for interactive recommendation. In *WSDM'20*, pages 816–824, 2020.
[138] Yu Lei, Zhitao Wang, Wenjie Li, Hongbin Pei, and Quanyu Dai. Social attentive deep q-networks for recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 2020.
[139] Xiangyu Zhao, Xudong Zheng, Xiwang Yang, Xiaobing Liu, and Jiliang Tang. Jointly learning to recommend and advertise. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3319–3327, 2020.
[140] Shenggong Ji, Zhaoyuan Wang, Tianrui Li, and Yu Zheng. Spatio-temporal feature fusion for dynamic taxi route recommendation via deep reinforcement learning. *Knowledge-Based Systems*, 205:106302, 2020.
[141] Peter Wei, Stephen Xia, Runfeng Chen, Jingyi Qian, Chong Li, and Xiaofan Jiang. A deep reinforcement learning based recommender system for occupant-driven energy optimization in commercial buildings. *IEEE Internet of Things Journal*, 2020.
[142] Yu Lei, Hongbin Pei, Hanqi Yan, and Wenjie Li. Reinforcement learning based recommendation with graph convolutional q-network. In *SIGIR'20*, pages 1757–1760, 2020.
[143] Dongyang Zhao, Liang Zhang, Bo Zhang, Lizhou Zheng, Yongjun Bao, and Weipeng Yan. Mahrl: Multi-goals abstraction based deep hierarchical reinforcement learning for recommendations. In *SIGIR'20*, pages 871–880, 2020.
[144] Feng Liu, Ruiming Tang, Xutao Li, Weinan Zhang, Yunming Ye, Haokun Chen, Huifeng Guo, and Yuzhou Zhang. Deep reinforcement learning based recommendation with explicit user-item interactions modeling. *arXiv preprint arXiv:1810.12027*, 2018.
[145] Feng Liu, Ruiming Tang, Xutao Li, Weinan Zhang, Yunming Ye, Haokun Chen, Huifeng Guo, Yuzhou Zhang, and Xiuqiang He. State representation modeling for deep reinforcement learning based recommendation. *Knowledge-Based Systems*, 205:106170, 2020.
[146] Weiwen Liu, Feng Liu, Ruiming Tang, Ben Liao, Guangyong Chen, and Pheng Ann Heng. Balancing between accuracy and fairness for interactive recommendation with reinforcement learning. *Advances in Knowledge Discovery and Data Mining*, 12084:155, 2020.
[147] Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. Estimation-action-reflection: Towards

deep interaction between conversational and recommender systems. In *WSDM'20*, pages 304–312, 2020.
[148] Xu He, Bo An, Yanghua Li, Haikai Chen, Rundong Wang, Xinrun Wang, Runsheng Yu, Xin Li, and Zhirong Wang. Learning to collaborate in multi-module recommendation via multi-agent reinforcement learning without communication. In *RecSys'20*, pages 210–219, 2020.
[149] Xiangyu Zhao, Long Xia, Lixin Zou, Hui Liu, Dawei Yin, and Jiliang Tang. Whole-chain recommendations. In *CIKM'20*, pages 1883–1891, 2020.
[150] Xiang Wang, Yaokun Xu, Xiangnan He, Yixin Cao, Meng Wang, and Tat-Seng Chua. Reinforced negative sampling over knowledge graph for recommendation. In *WWW'20*, pages 99–109, 2020.
[151] Xiaocong Chen, Chaoran Huang, Lina Yao, Xianzhi Wang, Wenjie Zhang, et al. Knowledge-guided deep reinforcement learning for interactive recommendation. In *IJCNN'20*, pages 1–8, 2020.
[152] G. Adomavicius, A.Tuzhilin. (2005) "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions", *IEEE Transaction on knowledge and data engineering*, VOL 17, NO.16, JUNE 2005.
[153] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl. "GroupLens: An open architecture for collaborative filtering of Netnews", *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, Chapel Hill, NC: Pages 175-186
[154] R. V. Meteren, M. V. Someren (2000). "Using Content-Based Filtering for Recommendation", *MLnet / ECML2000 Workshop*, May 2000, Barcelona, Spain.
[155] Sutton, R.S. and Barto, A.G. (1998) *Reinforcement Learning: An Introduction*, MIT Press, Cambridge.