SAURS: Self-Adaptive Unexpected Recommender System through Deep Q-Network

Pan Li
Stern School of Business
New York University
New York, NY 10012
pli2@stern.nyu.edu

Abstract

Unexpected recommender system constitutes an important method to tackle the problem of filter bubble and user boredom by providing satisfying and novel item recommendations simultaneously. Existing unexpected recommendation models typically utilize a hybrid utility function to jointly optimize relevance and unexpectedness objectives, while the balance between the two objectives are determined either subjectively by the engineers or through black-box models such as neural networks. In both cases, users have little control over the degree of unexpectedness they receive, while the user perception towards unexpectedness could shift drastically within each session. To address these challenges, we propose a novel Self-Adaptive Unexpected Recommender System (SAURS) that utilizes the reinforcement learning mechanism to capture the user dynamics. In particular, we determine the degree of unexpectedness incorporated for each recommendation through a Double Deep Q-Network that assesses the user preferences towards unexpectedness in real-time and update the unexpected recommendation model dynamically. Extensive offline experiments on three real-world datasets illustrate that the proposed SAURS model significantly outperforms the state-of-the-art baseline approaches in terms of both accuracy and unexpectedness measures. In addition, we conduct an online A/B test at a major video platform in China, where our model achieves significant improvements over latest production models in several business metrics.

1 Introduction

Most of the traditional recommender systems provide personalized recommendations to the users that are related to their past purchases. As a result, they lead to over-specialization biases Adamopoulos and Tuzhilin (2014), filter bubbles Pariser (2011); Nguyen et al. (2014) and user boredom Kapoor et al. (2015a,b), while not taking into account users' desire for the surprising and unexpected recommendations that would broaden their horizons. Those challenges negatively affect user online experiences, thus limiting usefulness and popularity of recommendation services.

To provide surprising *and* relevant recommendations, unexpected recommendation models typically adopt a hybrid utility function that jointly optimizes the relevance and unexpectedness objectives Adamopoulos and Tuzhilin (2015), where unexpectedness is defined as the distance between the newly recommended item and the items previously consumed by the user Li and Tuzhilin (2019, 2020); Li et al. (2020). By doing so, users would receive recommendations deviating from their prior expectations, which would allow them to break out of their filter bubbles, leading to significant increases in user satisfaction Chen et al. (2019). Several unexpected recommendation models have

Preprint. Under review.

been proposed in the literature Murakami et al. (2007); Zhang et al. (2012); Lu et al. (2012); Li et al. (2020), and some of them resulted in successful industrial deployment Li et al. (2020).

The key factor in providing successful unexpected recommendations lies in achieving the *optimal balance* between the relevance and the unexpectedness objectives Li and Tuzhilin (2020). However, achieving such optimal balance is not a simple task. On one hand, we do not want to provide recommendations that are too similar to the previous transactions made by the user for the aforementioned reasons. On the other hand, we do not want to present items that are too unexpected or even irrelevant to the users because they might be of little practical interest to them. Determining the optimal balance between relevance and unexpectedness is also important for business platforms to improve their business-oriented metrics, such as CTR (Click-Through-Rate) and GMV (Gross-Merchandise-Volume), and online user experiences at the same time, as argued in Li et al. (2020).

Although there have been some initial attempts made at identifying the balance between relevance and unexpectedness that was either subjectively provided in a manual fashion Adamopoulos and Tuzhilin (2015); Li and Tuzhilin (2020) or learned by a neural network model in a black-box way Li et al. (2020), there is no systematic method for determining such optimal balance in a comprehensive and rigorous manner for personalized recommendations. This is unfortunate because it is crucial for successful recommendations to achieve this "golden middle" balance in a personalized and dynamic fashion, so that the users would be in control by providing explicit or implicit feedback towards the right level of unexpectedness in their previous recommendations. For example, repeated boring familiar recommendations would increase users' desire for unexpected items, while we do not want to distract users with unexpected recommendations when they are watching their favorite TV series. In other words, it is crucial for successful unexpected recommender system to capture this user dynamics from their behaviors to properly assess user preferences towards unexpectedness in real-time and to dynamically update the recommendation model.

These observations motivate us to utilize *Reinforcement Learning (RL)* models Sutton and Barto (2018) to solve the optimal balancing task since they can get behavioral signals from the users in real-time and update their recommendation policies in a personalized and flexible manner over time. Further, the RL-based methods optimize long-term objectives, such as user engagement or user retention rates, thus aiming at identifying the optimal balance between relevance and unexpectedness over a long-time horizon and thus maximizing cumulative reward(s).

More specifically, in this paper we propose a novel *Self-Adaptive Unexpected Recommender System (SAURS)* model that learns dynamic user preferences from their interactions with candidate items and adapts the level of unexpectedness to be incorporated in the next item recommendation accordingly. In particular, the incorporated degree of unexpectedness is determined by a Double Deep Q-Network that searches for the optimal level of unexpectedness to maximize the long-term recommendation objectives. By doing so, we not only take into account personalized and session-based user interactions during the recommendation process, but also capture user dynamics from each individual action and adjust the next recommendation based on the current feedback in real time. We subsequently construct a hybrid utility function combining the degree of unexpectedness obtained from the reinforcement learning model with the click-through-rate predictions to produce final recommendations.

To demonstrate the benefits of the proposed method, we conduct extensive offline experiments on three real-world datasets, where the proposed SAURS model consistently and significantly outperforms all selected state-of-the-art baseline approaches in both usefulness and unexpectedness measures. We also conduct an online A/B test on a major video platform in China, where our model achieves significant improvements over the latest production model.

In this paper we make the following research contributions. We

- propose to adopt reinforcement learning mechanisms to provide *self-adaptive unexpected* recommendations;
- propose a novel SAURS model that implements this idea by monitoring and capturing changes in user preferences towards unexpectedness and dynamically adjusting recommendations to keep the right balance between relevance and unexpectedness in user preferences in real time;
- present extensive offline experiments and an online A/B test that demonstrate strengths of the proposed SAURS model.

The rest of the paper is organized as follows. We discuss the related work in Section 2 and present our proposed model in Section 3. Extensive offline experiments on three real-world dataset and four experimental settings are described in Section 4, while results as well as discussions are presented in Section 5. We present our online A/B test in Section 6. Finally, Section 7 summarizes our findings and concludes the paper.

2 Related Work

In this section, we discuss prior literature related to our work in two categories: unexpected recommendations and reinforcement learning-based recommendations.

2.1 Unexpected Recommendations

To tackle the challenges of over-specialization Adamopoulos and Tuzhilin (2014), filter bubbles Pariser (2011); Nguyen et al. (2014) and user boredom Kapoor et al. (2015a,b), researchers have proposed several beyond-accuracy metrics including unexpectedness, serendipity, novelty, diversity and coverage Murakami et al. (2007); Ge et al. (2010) for providing better recommendations. Although these metrics are closely related to each other, they are still vastly different in terms of their definition and formulation. For example, serendipity measures the subjective surprise feeling of the user towards the candidate item Chen et al. (2019); novelty measures the percentage of new recommendations that the users have not been exposed before McNee et al. (2006); diversity measures the variety of items as the pairwise similarity of items in the recommendation list Ziegler et al. (2005); and coverage measures the degree to which recommendations cover the set of available items Herlocker et al. (2004).

Unexpectedness, on the other hand, measures the deviation of the newly recommended item from the user's past behaviors. Different from the diversity measure, unexpectedness focuses on the similarity between the recommended item and the set of historic items, rather than the similarity with the recommendation list. Unexpectedness has attracted extensive research interests for its contribution towards user satisfaction Adamopoulos and Tuzhilin (2015); Li et al. (2020) and its ability to broaden user preferences Herlocker et al. (2004); Zheng et al. (2015). Therefore, various unexpected recommendation models has been proposed, including feature-based approaches Adamopoulos and Tuzhilin (2014, 2015) and deep learning-based approaches Li and Tuzhilin (2019, 2020). In particular, researchers in Li et al. (2020) have proposed the PURS model to provide personalized and session-based unexpected recommendations in an efficient manner, which leads to successful industrial deployment.

However, existing models do not provide a satisfying solution for intelligently balancing between the unexpectedness and usefulness objectives, as they are decided either manually Li and Tuzhilin (2019, 2020) or by black-box deep-learning models Li et al. (2020). In this paper, we propose to adopt reinforcement learning mechanism to dynamically identify the optimal level of unexpectedness for each recommendation, which significantly improves the recommendation performance.

2.2 Reinforcement Learning-Based Recommendations

Reinforcement learning Sutton and Barto (2018) provides a comprehensive understanding of how agents may optimize their control of an environment Mnih et al. (2015), and has been applied in many situations to approach real-world complexity. The success of reinforcement learning models is determined by its ability to derive efficient representations of the environment from high-dimensional data inputs, and use these representations to generalize past experience to new situations. Recent advantages in deep neural network has made it possible to conduct end-to-end training for reinforcement learning models, while the most representative work is Deep Q-Network (DQN) Mnih et al. (2015). To further improve the performance of reinforcement learning and tackle the problem of over-estimation, several variants of DQN have been proposed, including Double DQN Van Hasselt et al. (2015) and Duel DQN Wang et al. (2016).

Reinforcement learning-based approaches have also attracted a lot of attention in recommender systems, which aims at learning the optimal user actions to maximize the long-term objectives Zhao et al. (2018a). Common approaches include contextual bandit based methods Li et al. (2010); Qin et al. (2014); Wang et al. (2017), which handle the exploration-exploitation trade-off in the online

environment to solve the cold start problem; and Markov Decision Process (MDP) based methods Shani et al. (2005); Dulac-Arnold et al. (2015); Lu and Yang (2016); Zhao et al. (2018b); Zou et al. (2019); Zheng et al. (2018); Xin et al. (2020) that design various neural network structures to extract dynamic user preferences from interactive information. Note that it might lead to unsatisfying user experiences if we learn the recommendation policy directly online with real customers, most of existing methods use the historical interactions to train an offline policy, and therefore allow the recommender system to get over the blundering stage in the offline environment without creating unsatisfying user experiences. All of these prior models have achieved great success in optimizing for long-term objectives.

However, these models only apply reinforcement learning models to provide similarity or diversity based recommendations, without focusing on the important task of unexpected recommendations. In this paper, we propose to utilize the reinforcement learning framework to dynamically determine the optimal level of unexpectedness in each recommendation, rather than the recommendation itself, and achieve satisfying recommendation performance.

3 Model

3.1 Overview of the Recommendation Model

As previously discussed, the common practice for providing unexpected recommendations is to construct a hybrid utility function combining the usefulness and the unexpectedness measures. In addition, it is important to address heterogeneity of user perceptions towards unexpectedness by incorporating contextual information in the utility function. Based on this observation, we use the following hybrid utility function for user u and item i:

$$Utility_{u,i} = r_{u,i} + unexp\_factor_u * unexp_{u,i} \tag{1}$$

consisting of the following three components:

- *Click-Through-Rate Estimator* $\boldsymbol{r_{u,i}}$: it determines the degree of matching between the candidate item i and the interests of user u based on user features and past behaviors;
- *Perception of Unexpectedness* $\boldsymbol{unexp\_factor_u}$: it specifies perception of user u towards unexpectedness for the current recommendation in a dynamic and personalized manner;
- *Degree of Unexpectedness* $\boldsymbol{unexp_{u,i}}$: it specifies how much item i is unexpected for user u.

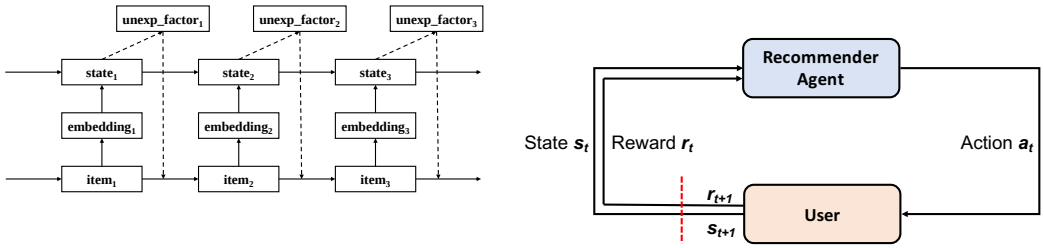
Perception of unexpectedness factor $unexp\_factor_u$ plays an important role in balancing between usefulness $\boldsymbol{r_{u,i}}$ and unexpectedness $\boldsymbol{unexp_{u,i}}$ components of the utility function and is crucial to the success of unexpected recommendations. Different from prior work that determines $unexp\_factor_u$ either manually Adamopoulos and Tuzhilin (2015); Li and Tuzhilin (2020) or through black-box neural network models Li et al. (2020), in this paper we adopt the reinforcement learning framework to *dynamically adjust* the value of $unexp\_factor_u$, as the user perception towards unexpectedness is changing dynamically during the online browsing session(s).

In particular, in this section we will introduce a new *Double Deep Q-Network framework* for determining $unexp\_factor_u$ by, first, formulating the problem and then constructing the state of the users based on latent user and item embeddings. We then describe the learning and optimization process of the proposed framework, as shown in Figure 1.

3.2 Problem Formulation

When user u sends a request to recommendation agent G, given the current state s_t of the user, our proposed method will select the optimal value of $unexp\_factor_u$ to be incorporated in the next recommendation. We model this problem as a Markov Decision Process (MDP) that includes a sequence of states, actions and rewards. More formally, MDP consists of the following five elements (S,A,P,R,γ):

- **State space** S: The user state representation $s_t \in S$ is modeled from the browsing history of the user and constructed in the latent space.



(a) SAURS Model (b) MDP

Figure 1: Overview of the proposed SAURS model and the MDP learning framework. The solid line represents sequential learning process and the dashed line represents reinforcement learning process

- **Action space** A: An action for the recommendation is to find the optimal level of unexpectedness to be included at time t based on the current user state s_t.
- **Reward** R: After the recommender agent G takes an action and determines user perception of unexpectedness $unexp\_factor_u$, we subsequently provide item recommendations to the user based on the hybrid utility function. The user can choose whether to click on the recommendations or not. The agent receives immediate reward $r(s_t, a_t)$ according to the user's feedback.
- **Transition probability** P: Transition probability P defines the probability of state transition when the recommender agent takes action a_t. The transition probability matrix is updated based on the observed reward.
- **Discount factor** γ: $\gamma \in [0, 1]$ defines the discount factor when we measure the present value of future reward. When $\gamma = 0$, the agent only considers the immediate reward. When $\gamma = 1$, all future rewards are fully taken into account in the current action. As a common practice, we choose γ as 0.99 in this paper.

With these definitions above, we formally define the problem of unexpected recommendations as follows: *Given the historical MDP (S,A,P,R,γ), the goal is to find the optimal policy $\pi : S \rightarrow A$ to determine the perception of unexpectedness, which can maximize the cumulative reward for the unexpected recommender system.* The MDP learning framework is also illustrated in Figure 1.

3.3 User and Item Embeddings

To effectively identify user interests and provide satisfying item recommendations, it is important to utilize explicit user and item features during the recommendation process. Motivated by the efficiency and effectiveness of deep-learning based models Zhang et al. (2019), we map the explicit feature information into the latent space and utilize autoencoding technique to obtain the latent embeddings. Specifically, we denote the explicit features for user u and item i as $W_u = [w_{u_1}, w_{u_2}, \cdots, w_{u_m}]$ and $W_i = [w_{i_1}, w_{i_2}, \cdots, w_{i_n}]$ respectively, where m and n stand for the dimensionality of user and item feature vectors. The goal is to train two separate neural networks: the encoder network that maps feature vectors into latent embeddings, and the decoder network that reconstructs feature vectors from latent embeddings. Due to effectiveness and efficiency of the training process, we represent both the encoder and the decoder as multi-layer perceptron (MLP). The MLP network learns the hidden representations by optimizing reconstruction loss L:

$$L = ||W_u - MLP^u_{dec}(MLP^u_{enc}(W_u))|| \tag{2}$$

$$L = ||W_i - MLP^i_{dec}(MLP^i_{enc}(W_i))|| \tag{3}$$

where MLP_{enc} and MLP_{dec} represent the MLP networks for the encoder and the decoder respectively. The MLP network is separately trained for obtaining user embedding and item embeddings to optimize for the reconstruction loss.

3.4 User State Representations

To construct an informative user state representation s_u that captures personalized, dynamic and contextual user preferences, we adopt the idea of sequence modeling and utilize GRU Cho et al. (2014) neural networks to condense user behavior sequence into latent embeddings. It is important to use the recurrent neural network to model user interests, for it is capable of capturing the time information and the sequence of user transactions, as more recent interactions would naturally have a higher impact on the current recommendation than previous interactions. In addition, compared with other recurrent neural network structures like RNN or LSTM, GRU is computationally more efficient and provides better performance Chung et al. (2014).

We denote the previous interaction sequence of user u as $P_u = [i_{u_1}, i_{u_2}, \cdots, i_{u_K}]$. During the learning process, we first map the behavior sequence to the corresponding item embeddings obtained in the previous stage. To illustrate the GRU learning procedure, we denote W_z,W_r,U_z and U_r as the weight matrices of current information and the past information for the update gate and the reset gate respectively. x_t is the user state input at timestep t, while h_t stands for the user state output. z_t denotes the update gate status and r_t represents the status of reset gate. Therefore, the hidden state at timestep t could be obtained following these equations:

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \tag{4}$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \tag{5}$$

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h) \tag{6}$$

Note that, each historic user behavior might have different influence on the current recommendation based on different context. Therefore, to capture the item-level heterogeneity in the behavior sequence, we propose to incorporate self-attentive mechanism Shaw et al. (2018) during the sequence modeling process. Typically, each output element o_t is computed as weighted sum of a linearly transformed input elements

$$o_t = \sum_{i=1}^{n} \alpha_{ti}(x_i W^t) \tag{7}$$

Each weight coefficient α_{ti} is computed using a softmax function

$$\alpha_{ti} = \frac{\exp e_{ti}}{\sum_{i=1}^{n} \exp e_{ti}} \tag{8}$$

where e_{ti} is computed using a compatibility function that compares two input elements x_t and x_i correspondingly.

By iteratively calculating hidden steps throughout the behavior sequence, we could obtain the hidden state at time t, which constitutes the representation of user state s_u for the current recommendation. The user state representation would be subsequently used for determining perception of unexpectedness as well as click-through-rate estimation.

3.5 Perception of Unexpectedness

Considering the aforementioned dynamic user perception of unexpectedness and the benefit to maximize future reward, we propose to use the Deep Q-Network (DQN) Mnih et al. (2015) framework to model the transition probabilities between different states to produce the optimal policy. As we discussed in the previous section, probability for a user to click on the current recommendation and future recommendations is the reward that the recommendation agent can get. The total reward is modeled as the following equation:

$$y_{s,a,t} = Q(s_t, a_t) = r_{immediate_t} + \gamma r_{future_t} \tag{9}$$

where state s is the user state representation learned from the historical interactions; action a is the level of user perception towards unexpectedness that balances between the unexpectedness and usefulness objectives; $r_{immediate_t}$ represents the immediate reward measuring whether user clicks on the recommendation at time t; r_{future_t} represents the agent's projection of future rewards; γ is the discount factor to balance between immediate rewards and future rewards. To tackle the common problem of over-estimation of q-value during the reinforcement learning process, we propose to

predict the total reward using the Double Deep Q-Network (DDQN) Van Hasselt et al. (2015). Given the current state s, the total reward is predicted as the following equation:

$$y_{s,a,t} = r_{a,t} + \gamma Q(s_{a,t+1}, argmax_{a'} Q(s_{a,t+1}, a'; W_t); W'_t) \tag{10}$$

where $r_{a,t}$ represents the immediate reward by taking action a. W_t and W'_t denote the sets of parameters of the target DQN and the primary DQN in the DDQN framework Van Hasselt et al. (2015). In the formulation, the recommender agent G speculates the next state $s_{a,t+1}$ given the selected action a. We aim to find the optimal action a' that gives the maximum future reward based on the primary DQN, and then estimate the future reward based on state $s_{a,t+1}$ using the target DQN. The parameters for the two DQN networks will be synchronized every few iterations. To provide an effective estimation of q-value, we follow the common practice Wang et al. (2016) to divide the Q-function into value function $V(s)$ and advantage function $A(s, a)$, where $V(s)$ is only determined by the current state, and $A(s, a)$ is determined by both the current state and the selected action. We formulate both the value and the advantage functions as Multi-Layer Perceptrons. The DQN model can be trained by minimizing the prediction error for the target q-value as:

$$Loss_{DQN} = E_{s_t, a_t, r_t, s_{t+1}}[(y_{s,a,t} - Q(s, a)]^2 \tag{11}$$

3.6 Click-Through-Rate Estimation

To provide the click-through-rate estimation, we concatenate the user state representation s_u with user embeddings W_u and item embeddings W_i generated in Section 3.3 and feed them into a MLP network to get the rating prediction:

$$r_{u,i} = MLP(W_u; s_u; W_i) \tag{12}$$

As we have discussed in Section 3.3, we utilize the self-attention mechanism Zhou et al. (2018) to determine the importance weight of each historic item by adaptively calculating the relevance metric towards current candidate recommendations.

3.7 Unexpectedness

In the prior literature, unexpectedness of a recommended item for a user is typically defined as the distance between the new item and the items previously consumed by the user, this distance being computed either in the feature space Adamopoulos and Tuzhilin (2015) or in the latent space Li and Tuzhilin (2019). As pointed out in Li and Tuzhilin (2020), latent modeling of unexpectedness achieves strong recommendation performance and improves novelty of recommendations without losing accuracy measures. Therefore we adopt a similar approach for modeling unexpectedness in this paper. Based on the user and item embeddings obtained in the previous section, we compute the degree of unexpectedness of item i for user u as the average Euclidean distance between the latent embedding of item i and the set of latent embeddings of all the previously consumed items by user u as:

$$unexp_{i,u} = \frac{1}{N} \sum_{k=1}^{N} d(W_i, W_k) \tag{13}$$

We also visualize the computation of unexpectedness measure in Figure 2, where the green dot stands for W_i and the red dots for W_k's. During the recommendation process, we jointly optimize the unexpectedness and the usefulness objectives, while keeping the right balance between the two by optimizing for the perception of unexpectedness factor $\boldsymbol{unexp\_factor_u}$ with the described reinforcement learning method.

3.8 Summary of the SAURS model

To summarize, we described in this section how to construct the hybrid utility function for providing unexpected recommendations by balancing between usefulness and unexpectedness objectives using a Double Deep Q-Network that captures dynamic user preferences. We also designed a self-attentive MLP network to model the user state and serve for the subsequent click-through-rate prediction task. The entire algorithm for SAURS is presented in Algorithm 1.

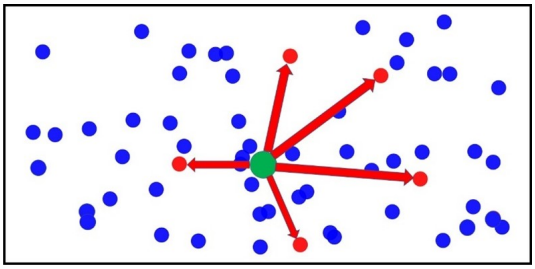


Figure 2: Visualization of unexpectedness modeling in the latent space. Blue points stand for the available items; Orange points represent the consumed items; Green point refers to the new recommended item.

Algorithm 1 Self-Adaptive Unexpected Recommender System

Input: User Feature u; Item Feature i; Historic User Interactions u_{past}; Click-Through-Rate y;
for each mini-batch **do**
    Obtain user and item embeddings W_u, W_i
    Calculate unexpectedness measure $unexp_{u,i}$ using Equation (13)
    Obtain user state representation s_u
    Obtain perception of unexpectedness $unexp\_factor_u$ using DDQN
    Obtain ctr prediction $r_{u,i} = MLP(W_u; s_u; W_i)$
    Construct utility function $y'_{u,i} = r_{u,i} + unexp\_factor_u * unexp_{u,i}$
    Backpropogate MSE loss $\hat{y} = y_{u,i} - y'_{u,i}$
    Backpropogate DQN loss $Loss_{DQN}$ using Equation (11)
    Synchronize DDQN
end for

4 Experiments

To empirically evaluate the performance of our proposed model, we conduct extensive offline experiments and compare it with several state-of-the-art baseline models. The codes are publicly available[1].

4.1 Data

We implement our model on three real-world industrial datasets: the Yelp Challenge Dataset [2] containing check-in information on users and restaurants, the MovieLens Dataset [3] containing information on user, movies and ratings, and the Youku dataset collected from the online video platform Alibaba-Youku, which contains rich information of users, videos, clicks and their corresponding features. We list the descriptive statistics of these datasets in Table 1. For the click-through rate prediction purposes, we binarize the ratings in Yelp and MovieLens datasets using the threshold level of 3.5 and convert them into the click vs. non-click labels.

Dataset	**Yelp**	**MovieLens**	**Youku**
# of Ratings	2,254,589	19,961,113	1,806,157
# of Users	76,564	138,493	46,143
# of Items	75,231	15,079	53,657
Sparsity	0.039%	0.956%	0.073%

Table 1: Descriptive Statistics of Three Datasets

[1] Codes are available at `https://anonymous.4open.science/r/f4e947d6-f204-43f0-97f3-cc4c27e9f2c4/`

[2] https://www.yelp.com/dataset/challenge

[3] https://grouplens.org/datasets/movielens/

4.2 Baseline Models

To illustrate that the proposed model indeed provide unexpected and useful recommendations at the same time, we select two groups of state-of-the-art baselines for comparison: five unexpected recommendation models and four click-through-rate prediction models. The first category includes:

- **PURS Li et al. (2020)** Personalized Unexpected Recommender System models the degree of unexpectedness in the latent space and provide personalized and session-based unexpected recommendations accordingly.
- **SPR Lu et al. (2012)** Serendipitous Personalized Ranking extends traditional personalized ranking methods by considering item popularity in AUC optimization.
- **Auralist Zhang et al. (2012)** Auralist is a personalized recommendation system that balances between the desired goals of accuracy, diversity, novelty and serendipity simultaneously.
- **DPP Chen et al. (2018)** The Determinantal Point Process utilizes a fast greedy MAP inference approach to generate relevant and diverse recommendations.
- **HOM-LIN Adamopoulos and Tuzhilin (2015)** HOM-LIN is the state-of-the-art unexpected recommendation algorithm, which provides recommendations through the hybrid utility function.

The second baseline category includes:

- **DIN Zhou et al. (2018)** Deep Interest Network designs a local activation unit to adaptively learn the representation of user interests from historical behaviors with respect to a certain item.
- **DeepFM Guo et al. (2017)** DeepFM combines the power of factorization machines for recommendation and deep learning for feature learning in a new neural network architecture.
- **Wide & Deep Cheng et al. (2016)** Wide & Deep utilizes the wide model to handle the manually designed cross product features, and the deep model to extract nonlinear relations among features.
- **PNN Qu et al. (2016)** Product-based Nerual Network model introduces an additional product layer to serve as the feature extractor.

4.3 Evaluation Metrics

To validate that the proposed model could provide unexpected and useful recommendations at the same time, we select the following popular accuracy and novelty metrics for the evaluation process: **AUC**, which measures the goodness of recommendation order by ranking all the items with predicted CTR and comparing with the click information; **HR@10**, which measures the number of clicks in top 10 recommendations; **Unexpectedness**, which measures the recommendations to users of those items that are not included in their consideration sets and depart from what they would expect from the recommender system; and **Coverage**, which measures as the percentage of distinct items in the recommendation over all distinct items in the dataset.

5 Results

5.1 Recommendation Performance

We applied the proposed SAURS model and the baseline methods described in the previous section to the aforementioned three real-world datasets. We applied time-stratified cross-validation with multiple random parameter initializations to this data and present the averaged performance results in this section. As shown in Table 2 and Figure 3, our proposed model consistently and significantly outperforms all other baselines in terms of both accuracy and unexpectedness measures across the three datasets. In comparison with the second-best baseline method, we observe an average increase of over 2% in AUC, 2% in HR@10, 7% in Unexpectedness and 1% in Coverage measures. We also observe that all the deep-learning based approaches perform significantly better than the feature-based methods, which demonstrates effectiveness of the latent models. We conclude that our proposed

Algorithm	Youku				Yelp				MovieLens			
	AUC	HR@10	Unexp	Coverage	AUC	HR@10	Unexp	Coverage	AUC	HR@10	Unexp	Coverage
SAURS	**0.7262***	**0.7608***	**0.0936***	**0.6174***	**0.6814***	**0.6809***	**0.2758***	**0.7634***	**0.8203***	**0.6917***	**0.2932***	**0.3788***
PURS	0.7154	0.7494	0.0913	0.6040	0.6723	0.6761	0.2401	0.7585	0.8090	0.6778	0.2719	0.3732
DIN	0.6957	0.6972	0.0688	0.1298	0.6694	0.6702	0.0391	0.6934	0.7021	0.6485	0.0887	0.2435
DeepFM	0.5519	0.5164	0.0333	0.2919	0.6396	0.6682	0.0412	0.6044	0.7056	0.6169	0.1275	0.3098
Wide&Deep	0.6807	0.6293	0.0472	0.3400	0.6698	0.6693	0.0392	0.7580	0.7940	0.6333	0.0944	0.3432
PNN	0.5801	0.5667	0.0593	0.1860	0.6664	0.6692	0.0391	0.7548	0.7140	0.6382	0.1318	0.3665
HOM-LIN	0.5812	0.5493	0.0602	0.4284	0.6287	0.6490	0.1433	0.5572	0.7177	0.5894	0.1116	0.1525
Auralist	0.5319	0.5250	0.0598	0.3990	0.6428	0.6104	0.1434	0.5442	0.6988	0.5710	0.1010	0.1333
SPR	0.5816	0.5156	0.0739	0.4668	0.6364	0.6492	0.1438	0.5849	0.7059	0.6122	0.1396	0.1728
DPP	0.6827	0.5777	0.0710	0.4702	0.5940	0.6414	0.1330	0.5072	0.7062	0.5984	0.1602	0.1908

Table 2: Comparison of recommendation performance in three datasets. The first block contains baselines for click-through-rate optimization, while the second block contains baselines for unexpectedness optimization. '*' represents statistical significance at the 0.95 level.

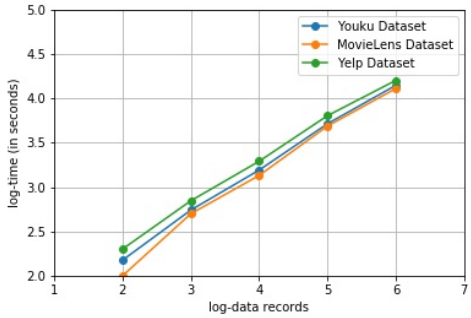


Figure 3: Scalability of SAURS on the Yelp, MovieLens and Youku datasets with increasing data sizes from 100 to 1,000,000 records.

approach achieves the state-of-the-art unexpected recommendation performance and indeed provides both useful *and* unexpected novel recommendations to the users.

In particular, it is important to observe that the proposed SAURS method significantly outperformed the previously proposed PURS approach that also defines the utility of recommendations as a mixture of usefulness and unexpectedness in the latent space, albeit without dynamically computing the Perception of Unexpectedness factor using the RL-based method described in this paper. Further, the improvements are statistically significant across all the three datasets and all the four performance metrics. This demonstrates the power and flexibility of the proposed SAURS approach and how skillfully reinforcement learning determines the "golden middle" between usefulness and unexpectedness.

5.2 Scalability

To test for scalability, we provide recommendations using SAURS with aforementioned parameter values for the Yelp, MovieLens and Youku datasets with increasing data sizes from 100 to 1,000,000 records. As shown in Figure 4, we empirically observe that the proposed SAURS model scales linearly with increase in number of data records to finish the training process and provide unexpected recommendations accordingly. The training procedure comprises of obtaining user and item embeddings and the reinforcement learning process. The optimization phase is made efficient using batch normalization Ioffe and Szegedy (2015) and distributed training. As our experiments confirm, SAURS is capable of learning network parameters efficiently and indeed scales well.

5.3 Ablation Study

As discussed in the previous section, the proposed SAURS model achieves significant improvements over other baselines, which demonstrates the power of reinforcement learning framework. To further justify the design of SAURS, we conduct the ablation study and compare the proposed model with the following variations:

- **SAURS-Variation 1 (DQN)** This model uses DQN, instead of Double DQN to predict the total reward in the reinforcement learning model.

Algorithm	Youku				Yelp				MovieLens			
	AUC	HR@10	Unexp	Coverage	AUC	HR@10	Unexp	Coverage	AUC	HR@10	Unexp	Coverage
SAURS	**0.7262***	**0.7608***	**0.0936***	**0.6174***	**0.6814***	**0.6809***	**0.2758***	**0.7634***	**0.8203***	**0.6917***	**0.2932***	**0.3788***
SAURS-Variation 1	0.7135	0.7292	0.0901	0.5848	0.6689	0.6710	0.2327	0.7580	0.8085	0.6739	0.2798	0.3580
SAURS-Variation 2	0.7024	0.7099	0.0875	0.5620	0.6691	0.6695	0.2102	0.7580	0.8072	0.6685	0.2671	0.3580
SAURS-Variation 3	0.7063	0.5628	0.0688	0.1298	0.6702	0.6702	0.0391	0.6934	0.7586	0.6331	0.0887	0.2435

Table 3: Ablation Study of recommendation performance in three datasets. '*' represents statistical significance at the 0.95 level.

	PV-CTR	VV-CTR	CTR	VV	TS	ID
Improvement	+4.48%*	+6.66%*	+4.98%*	+2.31%*	+3.13%*	+4.98%*

Table 4: Recommendation performance of SAURS in online A/B test: performance increase compared to the latest production model. '*' represents statistical significance at the 0.95 level.

- **SAURS-Variation 2 (No Future Reward)** This model sets the discount factor γ as 0 and only considers immediate reward during the recommendation process.
- **SAURS-Variation 3 (No Unexpectedness)** This model removes the unexpectedness component in the utility function, and provides recommendations based on click-through-rate estimations only.

As shown in Table 3, if we remove any of these four components out of the recommendation model, we will witness significant loss in both accuracy and unexpectedness measures, especially in coverage metric. Therefore, the ablation study demonstrates that the superiority of our proposed model really comes from the combination of four novel components that all play significant role in contributing to satisfying and unexpected recommendations.

6 Online A/B Test

We have also conducted the online A/B test at a major video platform in China during October 2020 in which we compared our SAURS model with the latest production model deployed by the company. We measured performance using six major business metrics used by the video platform: **PV-CTR** (Page-View Click-Through-Rate, the percentage of user clicks out of total page views); **VV-CTR** (Visit-View Click-Through-Rate, the percentage of user clicks out of total visit views); **CTR** (Click-Through-Rate, the percentage of user clicks out of total impressions), **VV** (Video View, average video views of each user); **TS** (Time Spent, average time that each user spends on the platform) and **ID** (Impression Depth, average impressions in one session). We present the performance results in Table 4, which shows significant and consistent improvements of SAURS over the current production model across all the six business metrics. These improvements stem from the unique application of reinforcement learning to our task of providing unexpected recommendations.

7 Conclusions

In this paper, we propose to apply reinforcement learning to provide personalized unexpected recommendations. In particular, we develope a novel method Self-Adaptive Unexpected Recommender System (SAURS) that implements this idea by constantly monitoring and capturing changes in user preferences towards unexpectedness and dynamically adjusting recommendations to keep the right balance between relevance and unexpectedness in user preferences in real time. This right degree of unexpectedness modeled as the *Perception of Unexpectedness* factor is determined by an RL-based Double Deep Q-Network that searches for the optimal value of this degree to maximize the long-term objectives of unexpected recommendations. This approach allows not only to take into account personalized and session-based user interactions during the recommendation process, but also to capture user dynamics from each individual action and adjust the next recommendation based on the current feedback in real time. This Perception of Unexpectedness factor is subsequently incorporated into the hybrid utility function as the weight aggregating the relevance and unexpectedness components of the utility function to produce the final recommendations.

To demonstrate the advantages of the SAURS approach, we conducted extensive offline experiments on three real-world datasets and showed that the proposed SAURS model consistently and significantly

outperformed all the state-of-the-art baseline approaches in terms of usefulness and unexpectedness measures. We also conducted an online A/B test on a major video platform in China, where our model achieved significant improvements over the current production model in terms of the PV-CTR, VV-CTR, CTR, VV, TS and ID business performance metrics.

Compared with the existing unexpected recommendation models, the proposed SAURS method has the following advantages. First, it is capable of capturing heterogeneous user preferences in real time and update the next recommendation dynamically. Second, like other reinforcement learning models, SAURS could identify the optimal recommendation policy that maximizes the long-term objectives, rather than dealing with the short-term objectives of recommendations. Finally, SAURS models generate significantly better unexpected recommendations, as demonstrated through extensive offline and online experiments.

As the future work, we plan to extend the off-policy learning framework of SAURS model to on-policy learning and make it more practical and efficient for industrial deployment. Also, we plan to discover better modeling of dynamic user interests to further improve unexpected recommendation performance.

References

Panagiotis Adamopoulos and Alexander Tuzhilin. 2014. 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*. ACM, 153–160.

Panagiotis Adamopoulos and Alexander Tuzhilin. 2015. On unexpectedness in recommender systems: Or how to better expect the unexpected. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5, 4 (2015), 54.

Li Chen, Yonghua Yang, Ningxia Wang, Keping Yang, and Quan Yuan. 2019. How Serendipity Improves User Satisfaction with Recommendations? A Large-Scale User Evaluation. In *The World Wide Web Conference*. ACM, 240–250.

Laming Chen, Guoxin Zhang, and Eric Zhou. 2018. Fast greedy MAP inference for Determinantal Point Process to improve recommendation diversity. In *Advances in Neural Information Processing Systems*. 5622–5633.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. ACM, 7–10.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. 2015. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679* (2015).

Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. 2010. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 257–260.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 5–53.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. 448–456.

Komal Kapoor, Vikas Kumar, Loren Terveen, Joseph A Konstan, and Paul Schrater. 2015a. I like to explore sometimes: Adapting to dynamic user novelty preferences. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 19–26.

Komal Kapoor, Karthik Subbian, Jaideep Srivastava, and Paul Schrater. 2015b. Just in time recommendations: Modeling the dynamics of boredom in activity streams. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. ACM, 233–242.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*. 661–670.

Pan Li, Maofei Que, Zhichao Jiang, Yao Hu, and Alexander Tuzhilin. 2020. PURS: Personalized Unexpected Recommender System for Improving User Satisfaction. In *Fourteenth ACM Conference on Recommender Systems*. 279–288.

Pan Li and Alexander Tuzhilin. 2019. Latent Modeling of Unexpectedness for Recommendations. *Proceedings of ACM RecSys 2019 Late-breaking Results* (2019), 7–10.

Pan Li and Alexander Tuzhilin. 2020. Latent Unexpected Recommendations. *Forthcoming at ACM Transactions on Intelligent Systems and Technology (TIST)* (2020).

Qiuxia Lu, Tianqi Chen, Weinan Zhang, Diyi Yang, and Yong Yu. 2012. Serendipitous personalized ranking for top-n recommendation. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2012 IEEE/WIC/ACM International Conferences on*, Vol. 1. IEEE, 258–265.

Zhongqi Lu and Qiang Yang. 2016. Partially observable markov decision process for recommender systems. *arXiv preprint arXiv:1608.07793* (2016).

Sean M McNee, John Riedl, and Joseph A Konstan. 2006. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI'06 extended abstracts on Human factors in computing systems*. 1097–1101.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

Tomoko Murakami, Koichiro Mori, and Ryohei Orihara. 2007. Metrics for evaluating the serendipity of recommendation lists. In *Annual conference of the Japanese society for artificial intelligence*. Springer, 40–46.

Tien T Nguyen, Pik-Mai Hui, F Maxwell Harper, Loren Terveen, and Joseph A Konstan. 2014. Exploring the filter bubble: the effect of using recommender systems on content diversity. In *Proceedings of the 23rd international conference on World wide web*. ACM, 677–686.

Eli Pariser. 2011. *The filter bubble: How the new personalized web is changing what we read and how we think*. Penguin.

Lijing Qin, Shouyuan Chen, and Xiaoyan Zhu. 2014. Contextual combinatorial bandit and its application on diversified online recommendation. In *Proceedings of the 2014 SIAM International Conference on Data Mining*. SIAM, 461–469.

Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based neural networks for user response prediction. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1149–1154.

Guy Shani, David Heckerman, and Ronen I Brafman. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6, Sep (2005), 1265–1295.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155* (2018).

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

Hado Van Hasselt, Arthur Guez, and David Silver. 2015. Deep reinforcement learning with double q-learning. *arXiv preprint arXiv:1509.06461* (2015).

Huazheng Wang, Qingyun Wu, and Hongning Wang. 2017. Factorization bandits for interactive recommendation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. 2695–2702.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. 2016. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*. 1995–2003.

Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M Jose. 2020. Self-Supervised Reinforcement Learning forRecommender Systems. *arXiv preprint arXiv:2006.05779* (2020).

Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 5.

Yuan Cao Zhang, Diarmuid Ó Séaghdha, Daniele Quercia, and Tamas Jambor. 2012. Auralist: introducing serendipity into music recommendation. In *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM, 13–22.

Xiangyu Zhao, Long Xia, Jiliang Tang, and Dawei Yin. 2018a. Reinforcement learning for online information seeking. *arXiv preprint arXiv:1812.07127* (2018).

Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018b. Deep reinforcement learning for page-wise recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 95–103.

Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. DRN: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 World Wide Web Conference*. 167–176.

Qianru Zheng, Chi-Kong Chan, and Horace HS Ip. 2015. An unexpectedness-augmented utility model for making serendipitous recommendation. In *Industrial Conference on Data Mining*. Springer, 216–230.

Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1059–1068.

Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*. ACM, 22–32.

Lixin Zou, Long Xia, Zhuoye Ding, Jiaxing Song, Weidong Liu, and Dawei Yin. 2019. Reinforcement Learning to Optimize Long-term User Engagement in Recommender Systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2810–2818.