PURS: Personalized Unexpected Recommender System for Improving User Satisfaction

Pan Li
New York University
New York, USA
pli2@stern.nyu.edu

Maofei Que
Alibaba Youku Cognitive and Intelligent Lab
Beijing, China
maofei.qmf@alibaba-inc.com

Zhichao Jiang
Alibaba Youku Cognitive and Intelligent Lab
Beijing, China
zhichao.jzc@alibaba-inc.com

Yao Hu
Alibaba Youku Cognitive and Intelligent Lab
Beijing, China
yaoohu@alibaba-inc.com

Alexander Tuzhilin
New York University
New York, USA
atuzhili@stern.nyu.edu

ABSTRACT

Classical recommender system methods typically face the filter bubble problem when users only receive recommendations of their familiar items, making them bored and dissatisfied. To address the filter bubble problem, unexpected recommendations have been proposed to recommend items significantly deviating from user's prior expectations and thus surprising them by presenting "fresh" and previously unexplored items to the users. In this paper, we describe a novel Personalized Unexpected Recommender System (PURS) model that incorporates unexpectedness into the recommendation process by providing multi-cluster modeling of user interests in the latent space and personalized unexpectedness via the self-attention mechanism and via selection of an appropriate unexpected activation function. Extensive offline experiments on three real-world datasets illustrate that the proposed PURS 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 Alibaba-Youku, where our model achieves over 3% increase in the average video view per user metric. The proposed model is in the process of being deployed by the company.

KEYWORDS

Recommender System, Unexpectedness, Personalization, Sequential Recommendation

ACM Reference Format:
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 (RecSys '20), September 22–26, 2020, Virtual Event, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3383313.3412238

RecSys '20, September 22–26, 2020, Virtual Event, Brazil

ACM ISBN 978-1-4503-7583-2/20/09…$15.00
https://doi.org/10.1145/3383313.3412238

1 INTRODUCTION

Recommender systems have been an important tool for online commerce platforms to assist users in filtering for the best content while shaping their interests at the same time. To achieve this objective, collaborative filtering has been among the most popular techniques and was successfully deployed for decades. However as shown in the recent literature, classical collaborative filtering algorithms often lead to the problem of over-specialization [2], filter bubbles [34, 35] and user boredom [25, 26]. To make things worse, users might get annoyed if they are recommended similar items repeatedly in a short period of time.

To address these issues, researchers have proposed to incorporate unexpectedness metric in recommendation models [33], the goal of which is to provide novel, surprising and satisfying recommendations. Unlike diversity, which only measures dispersion among recommended items, unexpectedness measures deviations of recommended items from user expectations and thus captures the concept of user surprise and allows recommender systems to break from the filter bubble. It has been shown in [1, 3] that unexpectedness leads to significant increase of user satisfaction. Therefore, researchers have introduced various recommendation algorithms to optimize the unexpectedness metric and achieved strong recommendation performance [3, 27, 29, 47].

However, very few of them have been successfully applied to industrial applications or achieved significant improvements in real business settings. This is the case for the following reasons. First, to improve unexpectedness of recommended items, previous models often have to sacrifice the business-oriented metrics [4, 51], such as CTR (Click-Through-Rate) and GMV (Gross-Merchandise-Volume). Second, most of the proposed unexpected recommendation algorithms are not scalable and therefore cannot be deployed in large-scale industrial applications. Third, the lack of personalization and session-based design of recommender systems reduces user satisfaction and their online browsing performance metrics. Thus, it is important to overcome these problems and intelligently deploy unexpected recommender systems in real-world applications.

Note that, while working on unexpected recommendations, it is crucial to address the problem of heterogeneous user preferences by focusing on providing personalized recommendations according to these preferences. For example, some people tend to be ''variety-seekers" [31] and are more willing to click on novel item recommendations, while others prefer to stay in their own comfort zones and are in favor of familiar item recommendations. In addition, even the same person might have different perceptions of unexpectedness in different contexts [5], which also motivates us to include session-based information into the design of an unexpected recommender system. For example, it is more reasonable to recommend the next episode of a TV series to the user who has just finished the first episode, instead of recommending new types of videos to that person. On the other hand, if the user has been binge-watching the same TV series in one night, it is better to recommend something different to him or her.

To address these concerns, in this paper we propose the *Personalized Unexpected Recommender System* (PURS) that incorporates unexpectedness into recommendations in an end-to-end scalable manner. Specifically, we model user interests as *clusters* of the previously consumed items in the *latent* space and subsequently calculate unexpectedness as the weighted distance between a new item and the clusters of these interests. Furthermore, we utilize sequence modeling and the self-attention mechanism to capture personalized and session-based user perception of unexpectedness. We also propose a novel unexpected activation function to adjust the network output for better unexpected recommendation performance. To provide unexpected recommendations, we construct a hybrid utility function by combining the aforementioned unexpectedness measure with estimated business performance metrics, especially the click-through-rate. Finally, extensive offline experiments on three real-world datasets illustrate that the proposed model consistently and significantly outperforms the state-of-the-art baseline approaches in both accuracy and novelty measures. We also conduct an online A/B test on a major video platform Alibaba-Youku, where our model achieves significant improvements over the current production model. Based on these positive results, the proposed model is in the process of being moved into production in the company.

This paper makes the following research contributions. It

(1) presents a novel PURS model that efficiently and effectively incorporates unexpectedness into recommendations;

(2) proposes to use the self-attention mechanism to model personalized and session-based user perception of unexpectedness;

(3) proposes to calculate unexpectedness as the distance between a new item and clusters of user interests in the latent space;

(4) presents a novel unexpected activation function to optimize performance of unexpected recommendations;

(5) presents extensive offline experiments and an online A/B test that empirically demonstrate the strengths of the PURS model.

2 RELATED WORK

In this section, we discuss prior literature related to our work in three categories: unexpected recommendations, deep-learning based recommendations and personalized & session-based recommendations.

2.1 Unexpected Recommendations

As discussed in the previous section, to overcome the problem of filter bubbles and user boredom, researchers have proposed to optimize beyond-accuracy objectives, including unexpectedness, serendipity, novelty, diversity and coverage [18, 33]. Note that, these metrics are closely related to each other, but still different in terms of definition and formulation. Specifically, serendipity measures the positive emotional response of the user about a previously unknown item and indicates how surprising these recommendations are to the target users [7, 40]; novelty measures the percentage of new recommendations that the users have not seen before or known about [32]; diversity measures the variety of items in a recommendation list, which is commonly modeled as the aggregate pairwise similarity of recommended items [52]; coverage measures the degree to which recommendations cover the set of available items [21].

Among them, unexpectedness has attracted particular research interests, for it is shown to be positively correlated with user experience [3, 7]. It also overcomes the overspecialization problem [3, 23], broadens user preferences [21, 47, 48] and increases user satisfaction [3, 29, 47]. Unexpectedness measures those recommendations that are not included in the users' previous purchases and depart from their expectations. Different from the diversity measure, unexpectedness is typically defined as the distance between the recommended item and the set of expected items in the feature space [3]. However, as pointed out in [27, 28], it is difficult to properly construct the distance function in the feature space, thus the authors propose to calculate the distance of item embeddings in the latent space instead.

Researchers have thus proposed multiple recommendation models to improve novelty measures, including Serendipitous Personalized Ranking (SPR) [29] that extends traditional personalized ranking methods by discounting item popularity in AUC optimization; Auralist [47] that balances the accuracy and novelty measures simultaneously using topic modeling; Determinantal Point Process (DPP) [8, 17] that utilizes the greedy MAP inference to diversify the recommendation results; HOM-LIN [3] that use a hybrid utility function of estimated ratings and unexpectedness to provide recommendations; and Latent Modeling of Unexpectedness [27, 28]. All of these models have successfully improved the unexpectedness measure of recommendations.

However, these prior literature aim to provide uniform unexpected recommendations to all users under all circumstances, without taking into account user-level heterogeneity and session-based information, thus might not reach the optimal recommendation performance. In addition, those models have the scalability issue and might not work well in the large-scale industrial settings. In this paper, we propose a novel deep-learning based unexpected recommender system to provide personalized and session-based unexpected recommendations in an efficient manner.

2.2 Deep-Learning based Recommendations

Another body of related work is around deep-learning based approaches to extract user preferences and provide recommendations. Compared with classical recommender systems, deep-learning based models are capable of conducting representation learning, sequence

modeling and nonlinear transformation to possess high level of flexibility [46]. Some representative work include [15, 42, 43] that construct heterogeneous information network embeddings to model complex heterogeneous relations between users and items; [20, 38] that apply deep neural network (DNN) techniques to obtain semantic-aware representations of users and items for efficient learning and recommendation; and [22, 39] that utilize autoencoder (AE) techniques to map the features of users and items into latent embeddings and model their relationships accordingly.

In this paper, we adopt deep-learning based approaches to model unexpectedness and subsequently provide unexpected recommendations, thus obtaining all these advantages discussed above.

2.3 Personalized and Session-based Recommendations

Note that different users might have different preferences towards the same recommendation, and even the same user might have different opinions towards the same recommendations on different sessions. Therefore, personalized and session-based recommender systems constitute important tools for providing satisfying recommendations. Specifically, these models take user behavior sequences into account to learn user behavior patterns and the shift of user preferences from one transaction to another [45]. Recent work on this field include DIN [50], DSIN [16], DIEN [49], DeepFM [19], Wide & Deep [9] and PNN [37] that combine user features, item features and user historic behaviors to provide useful recommendations.

In this paper, we incorporate personalized and session-based information into the model design to provide unexpected recommendations and improve user satisfaction.

3 UNEXPECTEDNESS

3.1 Modeling of Unexpectedness

In [3], the authors define unexpectedness of a recommended item as the distance between the new item and the closure of previous consumptions in the *feature* space. As discussed in the related work section, this approach might not achieve optimal performance, for the distance function is typically difficult to define in the feature space. Therefore in [27], unexpectedness is constructed in the *latent* space instead of the feature space, which is calculated as the euclidean distance to the entire latent closure of previous consumptions. This latent modeling approach of unexpectedness manages to achieve strong recommendation performance and improve novelty of recommendations without losing accuracy measures.

Note however, that many users have diverse interests, thus making it unreasonable to use one single closure of consumed items to model user interests. For example, in the movie recommendation applications, a user might have several different types of movie interests, including documentaries, fiction movies, comedies, animation and so on. Therefore, it is suboptimal to use one single closure to model wide interests and might be more appropriate to model user interests in separate clusters instead. In addition, as shown in Figure 1, if we take one single closure including all items that the user has consumed as the expected set, it might lead to the construction of an extremely large expected set and might accidentally include some unexpected items into the expected set as well. However, if we cluster the user's historic consumptions based on the diverse interests, it would be easier to identify the behavior patterns within each cluster and model the expectation sets accordingly. It also encourages those unexpected recommendations that could bridge the diverse interests of the same user, as illustrated in Figure 1.

Therefore, we propose to conduct clustering on embeddings of previous consumptions and form user interest clusters accordingly. We select the Mean Shift algorithm [10] to identify the clusters of historic behaviors automatically in the latent space for the following reasons. First, it is an unsupervised clustering algorithm, therefore we do not have to explicitly specify the number of interest clusters for each user, as we generally do not have that information as well. Mean Shift algorithm is capable of optimally selecting the best number of clusters without manual specification. Second, it is powerful for the analysis of a complex multi-modal feature space for recommendation applications, and it can also delineate arbitrarily shaped clusters in it [13].

Mean Shift clustering utilizes an iterative process to locate the maxima of a density function given discrete data sampled from that function. To handle the clustering procedure in our model, we denote the kernel function as $K(x_i - x)$ given an initial estimate x_i and observation x. This kernel function determines the weight of nearby points for re-estimation of the mean, which is typically modeled as a Gaussian distribution

$$K(x_i - x) = e^{-c||x_i - x||^2} \tag{1}$$

The weighted mean of the density in the window determined by K is calculated as

$$m(x) = \frac{\sum_{x_i \in N(x)} K(x_i - x) x_i}{\sum_{x_i \in N(x)} K(x_i - x)} \tag{2}$$

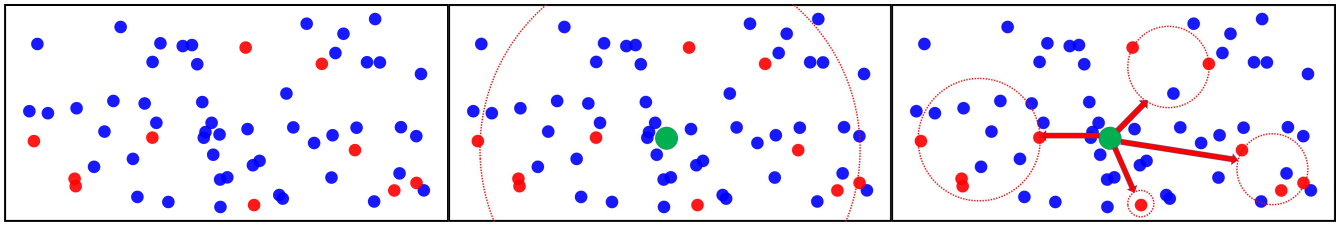
where $N(x)$ is the neighborhood of x. The mean-shift algorithm will reset x as $m(x)$, and repeat the estimation process until $m(x)$ converges.

For each user u, we extract the historic behavior sequence as $\{i_1, i_2, \cdots, i_n\}$ and their corresponding embeddings in the latent space as $\{w_1, w_2, \cdots, w_n\}$ through sequence modeling. We subsequently apply Mean Shift algorithm to cluster these embeddings into user interest clusters as $\{C_1, C_2, \cdots, C_N\}$. For each new item recommendation i_* for user u, unexpectedness is hereby modeled as the weighted average distance between each cluster C_k and embedding of the new item w_*:

$$unexp_{i_*,u} = \sum_{k=1}^{N} d(w_*, C_k) \times \frac{|C_k|}{\sum_{k=1}^{N} |C_k|} \tag{3}$$

3.2 Unexpected Activation Function

Though it is natural to directly include the unexpectedness obtained from the previous section into the utility function, it is suboptimal to do so, as our goal is to provide unexpected yet relevant and useful recommendations. If we explicitly combine unexpectedness into the hybrid utility function, it will tend to recommend items with very high unexpectedness, which are likely to be either irrelevant or even absurd to the user. In [3], the authors propose to use a unimodal function to adjust the unexpectedness input to the utility function.



(a) Item Embeddings (b) One Single Latent Closure (c) Clustering of Latent Closure

Figure 1: Comparison 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. We propose to define unexpectedness as the distance between the new item and clusters of latent closure generated by all previous consumptions.

However, unimodality alone is not enough for any function to serve as the unexpected activation function, as we also need to balance between unexpectedness and relevance objectives. For example, the unimodal Gaussian function does not obatin the optimal recommendation performance, as shown in Section 6. We consequently propose that the unexpected activation function $f(\cdot)$ should satisfy the following mathematical properties:

(1) **Continuity** For two items i_1 and i_2, if their unexpectedness towards user u are close enough ($|unexp_{u,i_1} - unexp_{u,i_2}| < \epsilon$), then their unexpectedness output to respective utility should also be close ($|f(unexp_{u,i_1}) - f(unexp_{u,i_2})| < \delta$), which implies the continuity requirement of $f(\cdot)$.
(2) **Boundedness** Note that the utility function $utility_{u,i}$ for any user u, item i should be bounded, therefore the corresponding unexpectedness output and the unexpected activation function should also be bounded ($|f(\cdot)| < \infty$). In addition, when unexpectedness goes to zero or infinity, it suggests the new item is either too similar to previous consumptions or totally irrelevant, thus its contribution to utility function should be negligible ($lim_{x\to 0} f(x) = 0$, $lim_{x\to\infty} f(x) = 0$).
(3) **Unimodality** For the optimization convenience, it is ideal to have a unimodal unexpectedness output to the utility function instead of multi-peaks which require additional effort for the recommendation model to balance between unexpectedness and relevance, as discussed in [3]. Therefore, we need to select a unimodal function as the activation function.
(4) **Short-Tailed** To provide unexpected yet relevant recommendations, it is important to note that relevance decreases very fast after unexpectedness increases above certain threshold, which indicates the use of a short-tailed or sub-Gaussian distribution [36]. Specifically, a sub-Gaussian distribution is a probability distribution with strong tail decay, whose tails are dominated by or decay at least as fast as the tails of a Gaussian. Therefore, the activation function should follow the sub-Gaussian distribution, the kurtosis of which should be less than 3 [6].

Many functions satisfy the properties above and thus become potential candidates for the unexpected activation function. To simplify the model and accelerate the optimization process, in this paper we choose a popular solution $f(x) = x * e^{-x}$ from the Gamma function [14] as the unexpected activation function, which satisfies all four required mathematical properties–it is a continuous, bounded and unimodal function with kurtosis value 1.5. As we show in Section 6.3, the unexpected activation function contributes significantly to the superior recommendation performance.

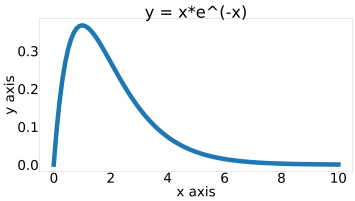


Figure 2: Unexpected Activation Function

4 PERSONALIZED UNEXPECTED RECOMMENDER SYSTEM

4.1 Overview

To provide unexpected recommendations, we propose to use the following hybrid utility function for user u and item i

$$Utility_{u,i} = r_{u,i} + f(unexp_{u,i}) * unexp\_factor_{u,i} \quad (4)$$

which consists of the following components:

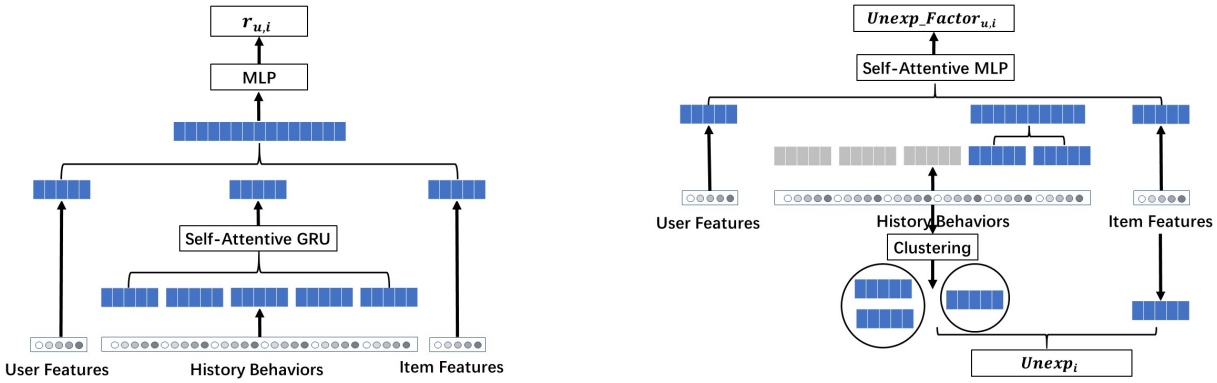
$\boldsymbol{r_{u,i}}$ that represents the click-through-rate estimation for user u towards item i based on their features and past behaviors.

$\boldsymbol{unexp_{u,i}}$ that represents the unexpectedness of item i towards user u, as introduced in the previous section.

$\boldsymbol{unexp\_factor_{u,i}}$ that represents the personalized and session-based perception of unexpectedness for user u towards item i.

$\boldsymbol{f(\cdot)}$ that stands for the activation function for unexpectedness in order to effectively and efficiently incorporate this piece into the utility function, as introduced in the previous section.

The proposed recommendation model is presented in Figure 3, which consists of two components: the base model, which estimates the click-through-rate of certain user-item pairs, as we will discuss in this section; and the unexpected model, which captures unexpectedness of the new recommendation as well as user perception towards unexpectedness, as we have discussed in the last section. The unexpected recommendations are provided through joint optimization of the utility function.



(a) Base (b) Unexpected

Figure 3: Overview of the proposed PURS model. The base model estimates the click-through-rate of certain user-item pairs, while the unexpected model captures unexpectedness of the new recommendation as well as user perception towards unexpectedness.

4.2 User and Item Embeddings

To effectively identify user interests and provide corresponding item recommendations, it is crucial to capture the feature-level information about the users and the items, which can be used for recommendation purposes in many different ways.

In addition, the intrinsic nature of users and items play an important role in determining the success of the recommendations. Some people, for example heavy users of the online video platform, tend to trust more on the recommendations provided by the platform. Thus for a certain recommendation for them, the utility function tend to be higher. On the other hand, the quality of an item is also crucial for the click-through-rate estimation, for the better quality of recommended items would lead to an increase of user satisfaction.

In this paper, we capture this user and item information in the form of *embeddings* in the latent space and utilize the deep-learning based autoencoder approach to obtain these user and item embeddings and also to capture their interactions in the latent space. 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 optimization reconstruction loss L:

$$L = ||W_u - MLP^u_{dec}(MLP^u_{enc}(W_u))|| \tag{5}$$

$$L = ||W_i - MLP^i_{dec}(MLP^i_{enc}(W_i))|| \tag{6}$$

where MLP_{enc} and MLP_{dec} represents the MLP network for encoder and decoder respectively. The MLP network is separately trained for obtaining user embedding and item embeddings, and is updated through back-propagation from the utility function optimization.

4.3 Click-Through-Rate Estimation ($r_{u,i}$) using Self-Attentive GRU

For a specific recommended item i, our goal is to predict whether user u would click on this recommendation or not, which largely depends on the matching between the content of item i and the interest of user u. This indicates the importance of precisely inferring user preferences from historic behaviors.

We denote the previous consumption of user u as the sequence $P_u = [i_{u_1}, i_{u_2}, \cdots, i_{u_K}]$ The click-through-rate prediction model obtains a fixed-length representation vector of user interests by pooling all the embedding vectors over the user behavior sequence P_u. We follow the idea of sequence modeling and utilize the bidirectional GRU [11] neural networks to obtain the sequence 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 order of user purchase, as more recent behavior would naturally have a higher impact on the current recommendation than previous actions. In addition, compared with other recurrent models like RNN or LSTM, GRU is computationally more efficient and better extracts semantic relationships [12].

During the training 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 behavior embedding input at timestep t, h_t stands for the output user interest vector, z_t denotes the update gate status and r_t represents the status of reset gate. The hidden state at timestep t could be obtained following these equations:

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \tag{7}$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \tag{8}$$

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h) \tag{9}$$

Note that, each historic consumption might have different influence on the current recommendation. For example, if a user have

watched the documentary "Deep Blue" from BBC and is very satisfied with the viewing experience, the user will be more likely to accept the current recommendation of documentary "Earth", also from BBC. Meanwhile, the historic record of watching James Bond might have relatively smaller influence towards the acceptance of that documentary recommendation. Therefore, to capture the item-level heterogeneity in the behavior sequence, we propose to incorporate self-attentive mechanism [41] during the sequence modeling process. Typically, each output element s_t is computed as weighted sum of a linearly transformed input elements

$$s_t = \sum_{i=1}^{n} \alpha_{ti}(x_i W^t) \tag{10}$$

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

$$\alpha_{ti} = \frac{\exp e_{ti}}{\sum_{i=1}^{n} \exp e_{ti}} \tag{11}$$

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

By iteratively calculating hidden step throughout every time step, we obtain final hidden state at the end of the behavior sequence, which constitutes the user interest embeddings R_u that captures the latent semantic information of the user's historic actions. To provide the click-through-rate estimation, we concatenate the obtained user interest embeddings R_u with user embeddings E_u and item embeddings E_i extracted in the previous section and feed into a MLP network to get the prediction: $r_{u,i} = MLP(R_u; E_u; E_i)$

4.4 Unexpected Factor ($unexp\_factor_{u,i}$) using Self-Attentive MLP

As we have discussed in the previous section, different people might have difference preferences towards unexpected recommendations, and their perception of unexpectedness is also influenced by the session-based information. Therefore, we need to take the user's historic actions in account when computing $unexp\_factor_{u,i}$ and provide personalized session-based recommendations.

We denote the previous consumption of user u as the sequence $P_u = [i_{u_1}, i_{u_2}, \cdots, i_{u_K}]$ Following the idea of session-based recommendation [30], instead of using the entire sequence to measure the factor of unexpectedness, we propose to only use a window of purchased items to identify the factor. The specific length of the window K is a hyperparameter that we can adjust to get the optimal performance.

The most recent user actions in the window will then be extracted and serve as the input to the MLP network. To capture the heterogeneity of the extracted historic behavior towards current unexpected recommendations, we utilize the structure of local activation unit [50] to determine whether the embedding of each item will be fed into the network. Instead of expressing the user's diverse interests with the same network structure, local activation unit is capable of adaptively calculating the relevance of historical behaviors towards current candidate recommendations.

Specifically, local activation unit performs a weighted sum pooling to adaptively calculate the activation stage of each behavior embedding and generate one single representation. We denote the sequence of item embeddings for user u in the session-window as $P_u = [E_{i_1}, E_{i_2}, \cdots, E_{i_K}]$ For user u and item i, the unexpected factor $unexp\_factor_{u,i}$ for this user-item pair will be calculated as

$$unexp\_factor_{u,i} = MLP(E_u; \sum_{j=1}^{K} a(E_u, E_{i_j}, E_i)E_{i_j}; E_i) \tag{12}$$

where $a(\cdot)$ is a feed-forward network with output as the activation weight for each past purchase.

5 EXPERIMENTS

In this section, we introduce extensive experiments that validate the superior recommendation performance of the proposed model in terms of both accuracy and novelty measures. The hyperparameters are optimized through Bayesian optimization [44]. For all experiments, we select $K = 10$ and use SGD as the optimizer with learning rate 1 and exponential decay 0.1. The dimensionality of embedding vector is 32. Layers of MLP is set by $32 \times 64 \times 1$. The batch size is set as 32. The codes are publicly available[1].

5.1 Data

We implement our model on three real-world datasets: the Yelp Challenge Dataset [2], which contains check-in information of users and restaurants; the MovieLens Dataset [3], which contains informations of user, movies and ratings; and the Youku dataset collected from the major online video platform 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 3.5 and transfer them into labels of click and non-click.

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

5.2 Baselines and Evaluation Metrics

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: click-through-rate prediction models and unexpected recommendation models. The first category includes:

- **DIN [50]** 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 [19]** DeepFM combines the power of factorization machines for recommendation and deep learning for feature learning in a new neural network architecture.

[1] Codes are available at https://github.com/lpworld/PURS
[2] https://www.yelp.com/dataset/challenge
[3] https://grouplens.org/datasets/movielens/

- **Wide & Deep [9]** 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 [37]** Product-based Nerual Network model introduces an additional product layer to serve as the feature extractor.

The second baseline category includes:

- **SPR [29]** Serendipitous Personalized Ranking extends traditional personalized ranking methods by considering item popularity in AUC optimization.
- **Auralist [47]** Auralist is a personalized recommendation system that balances between the desired goals of accuracy, diversity, novelty and serendipity simultaneously.
- **DPP [8]** The Determinantal Point Process utilizes a fast greedy MAP inference approach to generate relevant and diverse recommendations.
- **HOM-LIN [3]** HOM-LIN is the state-of-the-art unexpected recommendation algorithm, which provides recommendations through the hybrid utility function.

In addition, 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 is calculated as Equation (3); and **Coverage**, which measures as the percentage of distinct items in the recommendation over all distinct items in the dataset.

6 RESULTS

6.1 Recommendation Performance

We implement the proposed PURS model and baseline methods in three real-world datasets. We conduct the time-stratified cross-validation with different initializations and report the average results over these experiments. As shown in Table 2 and Figure 4, our proposed model consistently and significantly outperforms all other baselines in terms of both accuracy and unexpectedness measures across three datasets. Compare to the second-best baseline approach, we witness an increase of 2.75% in AUC, 6.97% in HR@10, 24.64% in Unexpectedness and 43.71% in Coverage measures. Especially in Youku Dataset where rich user behavior sequences in real business setting are available, PURS achieves the most significant improvement over other models. We also observe that all deep-learning based approaches performs significantly better than feature-based methods, which demonstrates the effectiveness of latent models. We conclude that our proposed approach achieves state-of-the-art unexpected recommendation performance and indeed provide satisfying and novel recommendations simultaneously to the target user.

6.2 Ablation Study

As discussed in the previous section, PURS achieves significant improvements over other baselines. These improvements indeed come from incorporating the following four components into the design of recommendation model: **Unexpectedness**, which aims at providing novel and satisfying recommendations; **Unexpected Activation Function**, which adjusts the input of unexpectedness into the utility function; **Personalized and Session-Based Factor**, which captures the user and session-level heterogeneity of perception towards unexpectedness; and **Clustering of Behavior Sequence**, which extracts the diverse user interests and constructs user expectations.

In this section, we conduct the ablation study to justify the importance of each factor. Specifically, we compare the proposed model with the following variations:

- **PURS-Variation 1 (Gaussian Activation)** This model users Gaussian distribution to serve as the unexpected activation function in the original design, and provides recommendations based on the following utility function $Utility_{u,i} = b_u + b_i + r_{u,i} + exp(-unexp^2_{u,i}) * unexp\_factor_{u,i}$
- **PURS-Variation 2 (No Activation)** This model removes the unexpected activation function in the original design, and provides recommendations based on the following utility function $Utility_{u,i} = b_u + b_i + r_{u,i} + unexp_{u,i} * unexp\_factor_{u,i}$
- **PURS-Variation 3 (No Unexpectedness Factor)** This model removes the unexpected activation function in the original design, and provides recommendations based on the following utility function $Utility_{u,i} = b_u + b_i + r_{u,i} + f(unexp_{u,i})$
- **PURS-Variation 4 (No Unexpectedness)** This model removes the unexpected activation function in the original design, and provides recommendations based on the following utility function $Utility_{u,i} = b_u + b_i + r_{u,i}$
- **PURS-Variation 5 (Single Closure of User Interest)** This model provides the unexpected recommendations using the same utility function, but instead remove the clustering procedure and model unexpectedness following [27] as $unexp_{i_*,u} = d(w_*, C_u)$ where C_u is the entire latent closure generated by all past transactions of user u.

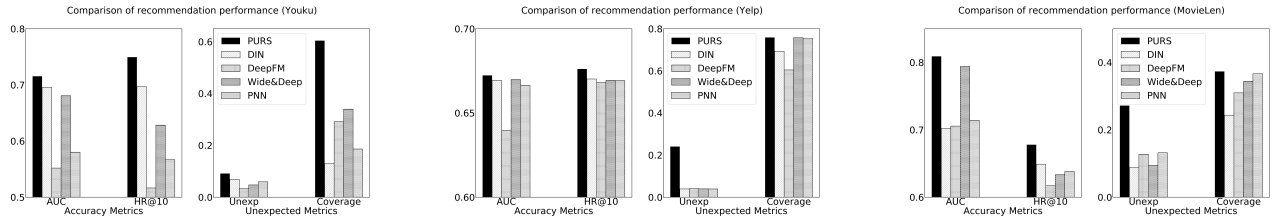
As shown in Table 3, if we remove any of these four components out of the recommendation model, we will witness significant loss in both accuracy and unexpectedness measures, especially in coverage metric. Therefore, the ablation study demonstrates that the superiority of our proposed model really comes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

6.3 Improving Accuracy and Novelty Simultaneously

In Table 2, we observe that unexpectedness-oriented baselines generally achieve better performance in unexpected measures, but at the cost of losing accuracy measures, when comparing to the CTR-oriented baselines. This observation is in line with the prior literature [4, 51] discussing the trade-off between these two objectives. However, our proposed PURS model manages to surpass baselines models in both **accuracy** and **novelty** measures at the same time. In addition, PURS is capable of improving AUC and unexpectedness metrics simultaneously throughout the training process, as shown in Figure 5.

Algorithm	Youku				Yelp				MovieLens			
	AUC	HR@10	Unexp	Coverage	AUC	HR@10	Unexp	Coverage	AUC	HR@10	Unexp	Coverage
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.



(a) Youku Dataset **(b) Yelp Dataset** **(c) MovieLen Dataset**

Figure 4: Comparison of recommendation performance in terms of accuracy and unexpectedness measures in three datasets.

Algorithm	Youku				Yelp				MovieLens			
	AUC	HR@10	Unexp	Coverage	AUC	HR@10	Unexp	Coverage	AUC	HR@10	Unexp	Coverage
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***
PURS-Variation 1	0.6826	0.7292	0.0828	0.5548	0.6682	0.6602	0.1298	0.7292	0.7757	0.6419	0.1812	0.3350
PURS-Variation 2	0.7067	0.7148	0.0707	0.3026	0.6692	0.6630	0.0412	0.7580	0.8041	0.6585	0.2471	0.3580
PURS-Variation 3	0.7036	0.6720	0.0762	0.1522	0.6508	0.6692	0.0391	0.7583	0.7666	0.6460	0.0888	0.2792
PURS-Variation 4	0.7063	0.5628	0.0688	0.1298	0.6702	0.6702	0.0391	0.6934	0.7586	0.6331	0.0887	0.2435
PURS-Variation 5	0.7038	0.7080	0.0477	0.2042	0.6701	0.6700	0.0395	0.7580	0.7715	0.6596	0.1727	0.3561

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

6.4 Scalability

To test for scalability, we provide recommendations using PURS 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 6, we empirically observe that the proposed PURS 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 jointly optimizing the utility function. The optimization phase is made efficient using batch normalization [24] and distributed training. As our experiments confirm, PURS is capable of learning network parameters efficiently and indeed scales well.

7 ONLINE A/B TEST

We conduct the online A/B test at Alibaba-Youku, a major video recommendation platform from 2019-11 to 2019-12. During the testing period, we compare the proposed PURS model with the latest production model in the company. We measure the performance using standard business metrics: **VV** (Video View, average video viewed by each user), **TS** (Time Spent, average time that each user spends on the platform), **ID** (Impression Depth, average impression through one session) and **CTR** (Click-Through-Rate, the percentage of user clicking on the recommended video). We also measure the novelty of the recommended videos using the unexpectedness and coverage measures described in Section 5.2. We present the results in Table 4 that demonstrates significant and consistent improvements over the current model in all the four business metrics, and these improvements indeed come from providing more unexpected recommendations, as demonstrated in Section

	VV	TS	ID	CTR	Unexpectedness	Coverage
Improvement	+3.74%*	+4.63%*	+4.13%*	+0.80%*	+9.74%*	+1.23%*

Table 4: Unexpected recommendation performance in online A/B test: performance increase compared to the current model. '*' represents statistical significance at the 0.95 level.

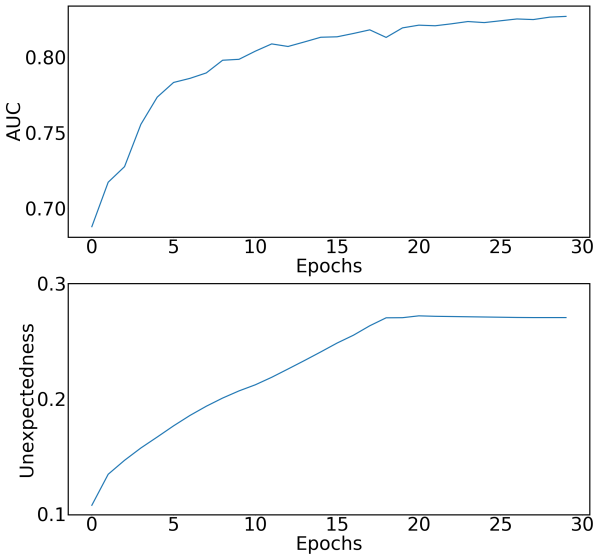


Figure 5: Improving accuracy and unexpectedness measures simultaneously in each training epoch of PURS in the MovieLens dataset

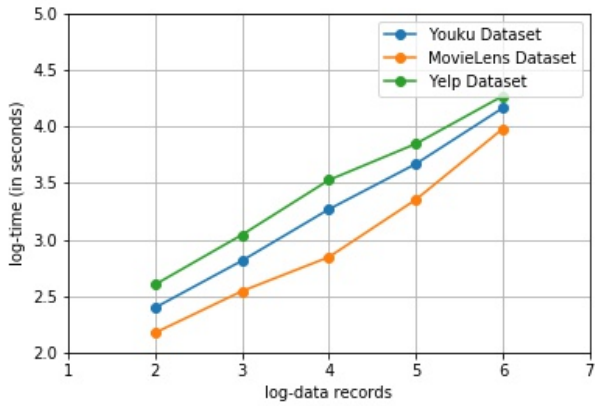


Figure 6: Scalability of PURS on the Yelp, MovieLens and Youku datasets with increasing data sizes from 100 to 1,000,000 records.

6. The proposed model is in the process of being deployed by the company.

8 CONCLUSIONS

In this paper, we present the PURS model that incorporates unexpectedness into the recommendation process. Specifically, we propose to model user interests as clusters of embedding closures in the latent space and calculate unexpectedness as the weighted distance between the new items and the interest clusters. We utilize the sequence modeling and the self-attention mechanism to capture personalized and session-based user perception of unexpectedness. We also propose a novel unexpected activation function to achieve better unexpected recommendation performance. We subsequently combine the CTR estimation with the degree of unexpectedness to provide final recommendations. Extensive offline and online experiments illustrate superiority of the proposed model.

As the future work, we plan to study the impact of unexpected recommendations on user behaviors. Also, we plan to model the user interests in a more explicit manner to provide better unexpected recommendations.

REFERENCES

[1] Panagiotis Adamopoulos. 2014. On discovering non-obvious recommendations: Using unexpectedness and neighborhood selection methods in collaborative filtering systems. In *Proceedings of the 7th ACM international conference on Web search and data mining*. ACM, 655–660.

[2] 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.

[3] 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.

[4] Gediminas Adomavicius and YoungOk Kwon. [n.d.]. Overcoming accuracy-diversity tradeoff in recommender systems: a variance-based approach. Citeseer.

[5] Gediminas Adomavicius and Alexander Tuzhilin. 2011. Context-aware recommender systems. In *Recommender systems handbook*. Springer, 217–253.

[6] Valerii V Buldygin and Yu V Kozachenko. 1980. Sub-Gaussian random variables. *Ukrainian Mathematical Journal* 32, 6 (1980), 483–489.

[7] 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.

[8] 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.

[9] 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.

[10] Yizong Cheng. 1995. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence* 17, 8 (1995), 790–799.

[11] 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).

[12] 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).

[13] Dorin Comaniciu and Peter Meer. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis & Machine Intelligence*

5 (2002), 603–619.
[14] Philip J Davis. 1959. Leonhard Euler's integral: A historical profile of the Gamma function: In memoriam: Milton Abramowitz. *The American Mathematical Monthly* 66, 10 (1959), 849–869.
[15] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 135–144.
[16] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep session interest network for click-through rate prediction. *arXiv preprint arXiv:1905.06482* (2019).
[17] Mike Gartrell, Ulrich Paquet, and Noam Koenigstein. 2017. Low-rank factorization of determinantal point processes. In *Thirty-First AAAI Conference on Artificial Intelligence*.
[18] 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.
[19] 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).
[20] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 173–182.
[21] 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.
[22] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *science* 313, 5786 (2006), 504–507.
[23] Leo Iaquinta, Marco de Gemmis, Pasquale Lops, Giovanni Semeraro, and Piero Molino. 2010. Can a recommender system induce serendipitous encounters? In *E-commerce*. InTech.
[24] 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.
[25] Komal Kapoor, Vikas Kumar, Loren Terveen, Joseph A Konstan, and Paul Schrater. 2015. I like to explore sometimes: Adapting to dynamic user novelty preferences. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 19–26.
[26] Komal Kapoor, Karthik Subbian, Jaideep Srivastava, and Paul Schrater. 2015. 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.
[27] Pan Li and Alexander Tuzhilin. 2019. Latent Modeling of Unexpectedness for Recommendations. *Proceedings of ACM RecSys 2019 Late-breaking Results* (2019), 7–10.
[28] Pan Li and Alexander Tuzhilin. 2020. Latent Unexpected Recommendations. *Forthcoming at ACM Transactions on Intelligent Systems and Technology (TIST)* (2020).
[29] 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.
[30] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction* 28, 4-5 (2018), 331–390.
[31] Leigh McAlister and Edgar Pessemier. 1982. Variety seeking behavior: An interdisciplinary review. *Journal of Consumer research* 9, 3 (1982), 311–322.
[32] 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.
[33] 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.
[34] 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.
[35] Eli Pariser. 2011. *The filter bubble: How the new personalized web is changing what we read and how we think*. Penguin.
[36] Yoon-Joo Park and Alexander Tuzhilin. 2008. The long tail of recommender systems and how to leverage it. In *Proceedings of the 2008 ACM conference on Recommender systems*. ACM, 11–18.
[37] 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.
[38] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. *Learning internal representations by error propagation*. Technical Report. California Univ San Diego La Jolla Inst for Cognitive Science.
[39] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, 111–112.
[40] Guy Shani and Asela Gunawardana. 2011. Evaluating recommendation systems. In *Recommender systems handbook*. Springer, 257–297.
[41] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155* (2018).
[42] Chuan Shi, Binbin Hu, Xin Zhao, and Philip Yu. 2018. Heterogeneous Information Network Embedding for Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2018).
[43] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. 2017. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering* 29, 1 (2017), 17–37.
[44] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*. 2951–2959.
[45] Shoujin Wang, Longbing Cao, and Yan Wang. 2019. A survey on session-based recommender systems. *arXiv preprint arXiv:1902.04864* (2019).
[46] 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.
[47] 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.
[48] 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.
[49] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5941–5948.
[50] 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.
[51] Tao Zhou, Zoltán Kuscsik, Jian-Guo Liu, Matúš Medo, Joseph Rushton Wakeling, and Yi-Cheng Zhang. 2010. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences* 107, 10 (2010), 4511–4515.
[52] 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.