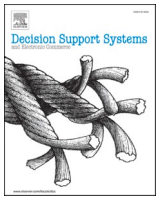
Decision Support Systems 169 (2023) 113957

Contents lists available at ScienceDirect

Decision Support Systems

journal homepage: www.elsevier.com/locate/dss



Live streaming recommendations based on dynamic representation learning

Ge Gao [a], Hongyan Liu [a,*], Kang Zhao [b]

[a] Research Center for Contemporary Management, Key Research Institute of Humanities and Social Sciences at Universities, School of Economics and Management, Tsinghua University, China
[b] Tippie College of Business, The University of Iowa, United States

ARTICLE INFO

Keywords:
Machine learning
Design science
Recommender systems
Consumer path

ABSTRACT

As an emerging form of social media, live streaming services (e.g., Twitch and Clubhouse) allow users to interact with hosts and peers in real time while enjoying shows or participating in discussions. These platforms are also dynamic, with shows or discussions changing quickly inside a room and users frequently switching between rooms. To improve user engagement and experience on such platforms, we design a new recommendation model named Dynamic Representations for Live Streaming Rooms (DRIVER) to provide room recommendations. Guided by the Integrated Framework for Consumer Path Modeling and the social affordance theory, DRIVER infers dynamic representations of live streaming rooms by leveraging users' behavior paths in entering, staying in, and leaving rooms. One contribution of our model is a new and efficient dynamic learning framework to model instantaneous and ever-changing inter-room relationships by considering individual users' behavior paths after leaving a room. Also supported by social affordance theory, another methodological novelty of our model is to capture dynamic characteristics of a room by incorporating features of the current audience inside the room. Experiments on real-world datasets from two different types of live streaming platforms demonstrate that DRIVER outperforms state-of-the-art representation learning methods and sequential recommender systems. The proposed method also has implications for recommender system design in other contexts, in which items are characterized by users' dynamic behavior paths and ongoing social interactions.

1. Introduction

Live streaming services represent a new type of online interactions and have experienced rapid growth during the past few years, with popular ones such as Twitch hosting nearly 150,000 rooms and attracting an average of more than two million concurrent users each day.[1] In live streaming, a host or streamer can start a room (i.e., a feed or channel on some platforms) to broadcast live shows to audience (e.g., Twitch and Tiktok) or initiate discussions among participants (e.g., Clubhouse). Unlike traditional social media (e.g., Twitter), TV programs with separate social media presence or asynchronous video streaming services (e.g., Youtube), the real-time social interactions among the audience afforded by live streaming platforms mean that the current audience inside a room is an indispensable part of characterizing the room. Meanwhile, because shows and discussions in rooms change quickly in live streaming, the audience can also leave a room for another room at any time. Such importance of a room's current audience and highly dynamic user behaviors distinguish live streaming from other forms of social media [1,2].

With tens of thousands of rooms streaming online simultaneously, it is imperative for live streaming platforms to recommend rooms to users to lower the search cost. Similar to recommender systems in social networks and eCommerce [3–5], accurate room recommendations have important business values for live streaming—they not only improve users' experience but also get hosts more engaged with the platform. However, existing recommendation methods have not fully considered nor leveraged the two key characteristics of live streaming we mention below.

First, user behaviors are highly dynamic, and so are rooms and inter-room relationships. Within a live streaming room, the host can stage many different shows over time, including shows of different types and styles, or even shows by guest performers. As a result, the characteristics of rooms are highly dynamic. At the same time, audience members also enter and leave rooms at a fast pace. For instance, in our real-world dataset from a live singing platform, each user stays in one room for a median of 2 min only and enters an average of 6.72 rooms during a day.

* Corresponding author at: School of Economics and Management, Tsinghua University, Beijing 100084, China.
E-mail addresses: gaog.18@sem.tsinghua.edu.cn (G. Gao), liuhy@sem.tsinghua.edu.cn (H. Liu), kang-zhao@uiowa.edu (K. Zhao).

[1] https://twitchtracker.com/statistics.

https://doi.org/10.1016/j.dss.2023.113957
Received 1 December 2021; Received in revised form 4 March 2023; Accepted 5 March 2023
Available online 15 March 2023

With such dynamic user behaviors in leaving and entering other rooms, the characteristics of a room should be frequently updated when users leave or enter the room. In contrast, most recommenders in e-Commerce are designed for items whose features barely change or change slowly.

Moreover, along with the dynamic behaviors of the hosts and audience, the sequential relationships among rooms are also dynamic. Because the shows and the audience in a room change fast, users who leave the same room at different time points may be attracted by different rooms and have different subsequent paths. Thus, the relationships among rooms based on user paths also change frequently. For instance, using one of the datasets used in our experiments from a live singing platform, we construct daily room-room graphs by connecting the room a user left with the room that this user subsequently entered. As shown in Fig. 1, the similarities of each room's relationships with other rooms between every two consecutive days are widely scattered[2]—the mean and median of such similarities are only 0.54 and 0.56, respectively, with a standard deviation of 0.28. This empirical evidence again highlights that inter-room relationships are dynamic over time, and such instantaneous connections among rooms may provide additional signals for room recommendations. However, while some studies captured sequential patterns of items via Recurrent Neural Networks (RNN) or constructed static graphs among items based on users' consumption paths and conducted graph learning with Graph Neural Networks (GNN), the literature offers no solution for how to model such instantaneous relationships among items, especially when item characteristics are also changing dynamically.

Second, the current audience inside a live streaming room plays an important role in characterizing the room. Through real-time social interactions, the audience in a room can influence not only peers but also the host. For one thing, online chats, comments, and gifts from audience members within a room can often be engaging and sometimes even become a major attraction of the room to other users [6]. For another, the effect of social interactions among audience members in a room goes beyond the audience alone and may have a direct and real-time impact on the performer and consequently the shows [2]. For example, a host may change the content or style of her show after getting feedback from the audience. Therefore, to characterize a live streaming room, we should consider both features of the room itself and features of its current audience. Nevertheless, while some recommenders treat all the users who have interacted with an item (e.g., shoppers who bought a product) in the past as equally important [7–9], none has explicitly emphasized the value of users who are currently interacting with an item (i.e., the audience inside a live streaming room).

To incorporate these characteristics and address the challenges associated with them, we propose a new recommendation model to learn Dynamic Representations for Live Streaming Rooms (DRIVER). Guided by the Integrated Framework for Consumer Path Modeling [10] and social affordance theory [11], DRIVER takes advantage of users' paths in entering, staying in, and leaving rooms to better capture dynamic characteristics of users, rooms, and inter-room relationships.

To evaluate the performance of the proposed model, we conduct comprehensive experiments on real-world datasets from two different types of live streaming platforms. Comparisons with other state-of-the-art recommendation models demonstrate that DRIVER provides the best and robust performance in live streaming room recommendation. Ablation studies reveal that both new components of DRIVER contribute to the improved recommendation performance.

The novelty and contributions of this paper can be summarized as follows. *First*, we propose DRIVER, a novel recommendation model for live streaming platforms. The model's general framework can be directly adopted by different live streaming platforms, who can train the model with their own data for recommendation tasks in their own services. Compared to existing methods, DRIVER has the following merits.

- DRIVER captures inter-room relationships based on users' behavior paths in departing from one room and joining another, and it incorporates such relationships into user-specific room recommendations. Different from existing sequential learning methods based on static item relationships or graph neural network models that are computationally expensive, our approach represents a new method for effectively and efficiently utilizing this kind of dynamic item graph to improve recommendation performance.
- DRIVER explicitly models the audience currently in a room as the room's instantaneous features to better capture the room's social characteristics that influence users' decisions. The model incorporates such instantaneous features with the room's cumulative features to obtain a comprehensive and user-specific representation of each room. This is the first effort to learn the dynamic representation of an item by considering users who are currently interacting with the item. Such representations are also updated when a new user-item interaction occurs.

Second, through extensive experiments, we demonstrate the performance gain of DRIVER over state-of-the-art recommendation models. More importantly, we reveal the unique value of DRIVER's design principles–learning dynamic representations of items based on users' behavior paths and current users' characteristics–in supporting users' decision-making in highly dynamic and social scenarios, such as live streaming.

The remainder of this paper is organized as follows. Section 2 reviews previous studies related to our work as well as the empirical evidence and theories that guide our design. Section 3 formalizes the live streaming room recommendation problem and describes the architecture of DRIVER. This is followed by a presentation of the experiment setup and results in Section 4. Then, Section 5 explores models learned by DRIVER and discusses why DRIVER works. Finally, Section 6 concludes this paper and discusses future research directions.

2. Background and related work

This section discusses related research to highlight the foundation and novelty of the proposed DRIVER model. We first review recent developments in machine learning methods for recommendation systems, including (dynamic) representation learning, sequential recommenders, and graph learning. After discussing their limitations in relation to the task of live streaming recommendations, we then introduce the Integrated Framework for Consumer Path Modeling and social affordance theory and explain how they guide our model design.

2.1. Representation learning for recommendations

In recent years, machine learning, especially deep learning, has attracted great research interest and shown superior performance on many problems, including recommendation systems [12]. As an important and fruitful area of machine learning research, representation learning [13] has been widely adopted for mining graph data [14]. In the context of recommendation systems, representation learning based models usually represent a user or an item with a latent vector to depict the user's preference or the item's characteristics. This method allows us to learn latent vectors (a.k.a., representations) for users and items to represent their characteristics from only their historical interactions. That means we can learn features of users and items without any explicit features of them other than using IDs of users and rooms IDs as inputs. Note that we use the terms "features" and "characteristics" interchangeably in this paper.

[2] Each daily room-room graph is a weighted directed graph. The weight of the edge from room A to room B is the number of users leaving A and entering B on that given day. The similarity of one room's relationships with others between two consecutive days is the Cosine similarity between this room's vectors from adjacency matrices of the two daily graphs.

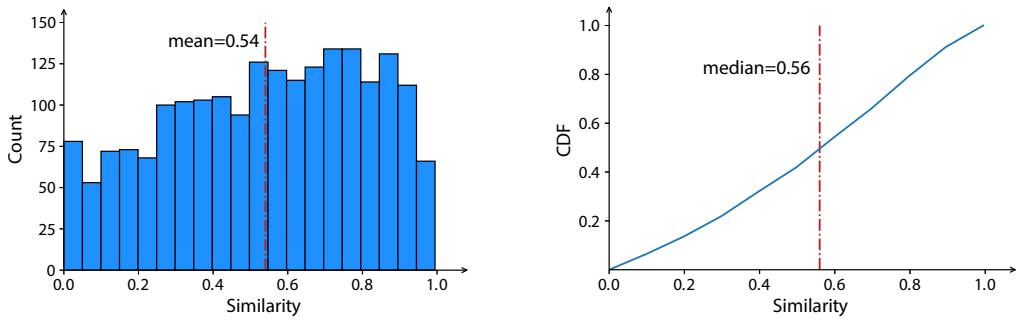


Fig. 1. The (a) histogram and (b) CDF (Cumulative Distribution Function) of similarities of inter-room relationship matrices between two consecutive days in the live singing platform dataset.

Traditional recommendation systems based on representation learning usually take users or items as static entities and learn static representations for them, such as the approach used in the classic Matrix Factorization model [7]. Although other recommenders have attempted to simulate the interactions between users and items using more complex methods based on their representations, they are based on the same assumption that these representations are static. However, static representations are not sufficient when user interests or item characteristics change over time, such as in social media and online social networks [15]. Therefore, dynamic representation learning methods have been developed using two types of models: probabilistic models [16] and RNN-based models [9]. The basic ideas of these two types of models are similar: representations for users and items co-evolve when interactions occur between them. There are also differences between them: Probabilistic models usually make some assumptions about the distribution of user-item interaction occurrence time and leverage these representations to calculate parameters of the distribution. In contrast, RNN-based models learn representations to calculate the probability of interaction occurrences directly. The latter usually shows better performance. For instance, the recently proposed model JODIE [9] achieves better performance by adopting RNN to update users' and items' representations and using a projection operation to model the temporal drift of these representations.

Nevertheless, these approaches only model a user-item interaction event as a one-time occurrence (e.g., purchasing a product) while ignoring users' subsequent behaviors after the start of the event. Specifically, for live streaming rooms, these methods would only consider users' entrance into a room. Thus, existing dynamic representation learning methods have two limitations when applied to live streaming recommendations: (1) they do not track users who are currently staying inside a room, and (2) they do not consider users' behaviors after leaving a room.

2.2. Sequential recommenders and graph neural networks

One stream of research that can potentially address the second limitation mentioned above is related to sequential recommendation systems. Such recommenders assume that the order of users' interactions with items matters and leverage sequential patterns of such behaviors for item recommendations [17]. Existing sequential recommenders learn from sequential data with Markov Chain (MC) models or deep learning models. Models based on MC usually estimate the transition probability between items from users' historical behavior sequences [18], but usually makes a strong Markov assumption that only the immediately last choice of a user directly determines her next action. Among deep learning methods for sequential recommendations, RNN-based models [9] use recurrent neural networks to handle the item sequence and capture user's preference. CNN-based models [19] take advantage of convolutional operations to extract features from users' historical behavior sequences. However, existing sequential recommenders are designed with the assumption that users and items have representations that are static or change very slowly. Thus, they cannot be directly applied to dynamic scenarios such as live streaming, where both users and rooms have characteristics that change quickly over time.

Another method for mining inter-item relationships is Graph Neural Networks (GNN) [20], which can learn representations for a node in a graph by recursively integrating its neighbors' features. Existing recommendation models based on GNN have been applied to item co-occurrence graphs [21], user-item bipartite graphs [8], and social networks among users [22]. However, GNN-based recommenders would face significant computational challenges if structures of underlying graphs change very frequently (e.g., on a minute basis, as in inter-room graphs for live streaming). On one hand, transductive GNN models, whether they are based on walks [23] or GCNs [24], are learned for a specific graph and have to be re-learned whenever graph structures change. On the other hand, although inductive GNN models [25] learn a set of models or aggregation functions to generate node embeddings, they still have low training efficiencies when the graph structure changes at the pace of inter-room graphs in live streaming.

2.3. Empirical and theoretical support for DRIVER

Our literature survey reveals that no existing recommender system accommodates the two key characteristics of live streaming—dynamic user behaviors (entering and leaving rooms) and social interactions—at the same time. As a new recommendation system for live streaming rooms, DRIVER integrates different machine learning methods in a novel way, and develops new components guided by empirical findings of consumer behaviors and social affordance theory.

In the marketing literature, data that record a consumer's spatial movement, whether it is in a physical or virtual space, are referred to as "path data". Based on empirical findings from research on consumer path data, the Integrated Framework for Consumer Path Modeling [10] is proposed for this emerging area of research. According to the different types of movements defined by the framework based on spatial configurations, the paths of audience members (i.e., consumers of digital content) in live streaming can be considered as non-physical and discrete movements with a low degree of constraints, as users can move freely between individual rooms in the virtual space. Therefore, we adopted the framework to guide our design of DRIVER.

First, the framework highlighted that path data help to better understand consumption behaviors [26], such as consumers' paths to purchases [27], so that business decisions can be optimized (e.g., store layout or product recommendations) [28]. Motivated by the value of consumer path data from empirical studies, DRIVER focuses on the dynamic paths of the audience when entering and leaving rooms and uses such path data in two ways: (1) It integrates JODIE [9] to update the embeddings of both users and rooms after users' entrance into rooms.

This allows the representations of users and rooms to stay up to date when user behaviors in live streaming are highly dynamic; (2) Extending the idea of sequential recommenders and graph learning, it proposes a novel and efficient way to capture dynamic inter-room relationships based on user paths of joining another room after leaving one room.

Second, this framework also identified that social interactions among individuals will affect their decisions of next movements [29]. DRIVER's emphasis on modeling the current audience inside a room is particularly supported by such importance of social interactions. As a unique feature of live streaming compared to other social media, real-time social interactions play important roles in improving user engagement, attracting more users, and improving revenue for live streaming platforms [30–32]. Many hosts also adjust their shows based on their audiences' real-time requests. Consequently, social interactions within a room can affect the content generation process or even determine the content or style of ongoing shows [2].

In fact, the value of social interactions for social media in general has been supported by social affordance theory [11], which measures the social capabilities provided by information systems. One key dimension of social affordance is social interactivity, which affords the possibility for users to interact with each other. Research has found that many users join live streaming because of social interactivity [33,34], and hosts that encourage social interactivity tend to be more popular [35].

Because interaction mechanisms differ in live streaming platforms, directly capturing social interactions in a room would require the analysis of text, image, voice, and/or video data using various computationally expensive methods. Also, when shows are different (e.g., gaming vs singing), the way users interact and how their interaction should be analyzed would vary as well. Therefore, DRIVER takes an indirect yet much more generalizable approach to reflect the importance of social interactivity and social affordance in live streaming—it aggregates features of current audience in a room into the room's instantaneous features, which are part of the room's comprehensive representation. Along with dynamic room representations based on all users who have visited the room in the past, such an aggregation explicitly captures the instantaneous social atmosphere within a room. In addition, this new emphasis on the current audience also ensures that users who spend more time in a room would have more chances to contribute to the room's characteristics than users who come and go quickly. Such an approach can be easily adopted by most live streaming platforms that allow real-time user interactions inside rooms.

Last, the framework also makes it clear that consumers are heterogeneous, and so are their path decisions [36]. In online streaming, this means that users may choose a room for different reasons. As a result, DRIVER attempts to address such user heterogeneity by generating user-specific room representations.

In sum, DRIVER dynamically updates the representations of users and items upon users' entrance into a room, explicitly captures social interactions inside a room with its current audience, and efficiently models the ever-changing inter-room relationships based on users' new destinations after leaving a room. The next section describes the model in detail.

3. The proposed method

This section first present the intuition behind DRIVER with a toy example and then provides a formal definition of the live streaming recommendation problem. Finally, we introduce the framework of the proposed DRIVER model, followed by detailed descriptions of each module.

3.1. A toy example

Suppose a live streaming platform has two rooms 1 and 2. At time t_1, user A, B, C are in room 1 and user D, E are in room 2. Then user A leaves room 1 and enters room 2, such that at time t_2 ($t_2 > t_1$), room 1 has users B and C, and room 2 has users A, D, and E. Fig. 2 depicts this example. Then we can infer that (1) room 2 is able to attract user A because of ongoing shows or the social atmosphere insider the room; and (2) there exists certain relationship between the two rooms, such that other users like A may also follow the same path (i.e., join room 2 after leaving room 1).

To capture such information, we use latent vectors to represent each user, room, and room's relationships with others. When a user leaves one room and enters another, we update the representations for users and rooms in three steps. For this toy example, after A joins room 2, we first update user A's vector by incorporating the vectors of room 2, user D and user E, so that user A's vector reflects the room that she likes and users she wants to interact with. Then room 2's vector is updated with user A's vector, so that the room's vector represents users who enjoy it. Finally, vectors of user A and room 2 are incorporated into room 1's relationship vector, so that the vector can represent the types of users who would join room 2 after leaving room 1.

3.2. Notations and the problem definition

In a live streaming platform with user set $\boldsymbol{U}$ and room set $\boldsymbol{M}$, our model would learn latent vectors (i.e. representations) for each user $u \in \boldsymbol{U}$ and each room $r \in \boldsymbol{M}$. Similar to other representation learning methods, these latent vectors will be learned from the data and no explicit features are required. Such a setup would only require users' behaviors in entering and leaving rooms as the input data for the model. This greatly improves the generalizability of the model, because such behaviors are universal in almost all live streaming platforms, despite the great variations in their streaming mechanisms and the types of shows.

Representations for users include static features and dynamic features. We use one-hot vector $\overline{\boldsymbol{u}} \in \boldsymbol{R}^{|U|}$ to denote u's static features that do not change over time. Let a d-dimension vector $\boldsymbol{u}(t)$ denote the user's dynamic features at time t, which is randomly initialized and will be learned through our proposed model.

Similarly, room r's static features are also denoted as one-hot vector $\overline{\boldsymbol{r}} \in \boldsymbol{R}^{|M|}$. Different from users, a room's dynamic features include two parts: (1) user-specific comprehensive features $\boldsymbol{s}_u^r(t)$ at time t for user u. It incorporates the room's cumulative features since its inception (denoted as $\boldsymbol{r}^c(t)$), the room's instantaneous features (denoted as $\boldsymbol{r}^i(t)$) reflected by $\boldsymbol{U}_r^t$, the set of users who are currently in room r, and user u's dynamic features $\boldsymbol{u}(t)$. (2) room's relationship features $\boldsymbol{r}^e(t)$ that captures room r's dynamic relationships with other rooms at time t.

Interactions between users and rooms are represented as temporal sequences of tuples $enter(u,r,t)$, indicating user u's entrance into room r at time t, and $leave(u,r,t)$, representing user u's departure from room r at time t. By tracing $enter(u,r,t)$ and $leave(u,r,t)$, we can get $\boldsymbol{U}_r^t$, the set of users in room r at time t.

Suppose user u leaves room r at time t and is ready to enter another room at time $t + \Delta t$, DRIVER will predict which room u will enter. Given a sequence of user-room interactions $enter(u,r,t)$ and $leave(u,r,t)$ until time t, the model first predicts $\widehat{\boldsymbol{s}}_u(t + \Delta t)$, the comprehensive features of the room that user u is most likely to enter at time $t + \Delta t$. Then, top K rooms whose comprehensive features right before $t + \Delta t$ are most similar to $\widehat{\boldsymbol{s}}_u(t + \Delta t)$ will form the recommendation list to user u. Table 1 summarizes all the notations used in this paper and more explanations can be found in Appendix A.

3.3. The general framework

Fig. 3 shows the general framework of DRIVER and pseudo-code is provided in Appendix A. Overall, the model consists of two main modules: the Representation Learning Module and the Predicting Module.

The Representation Learning Module consists of the Integrating Component and the Updating Component. This module handles the

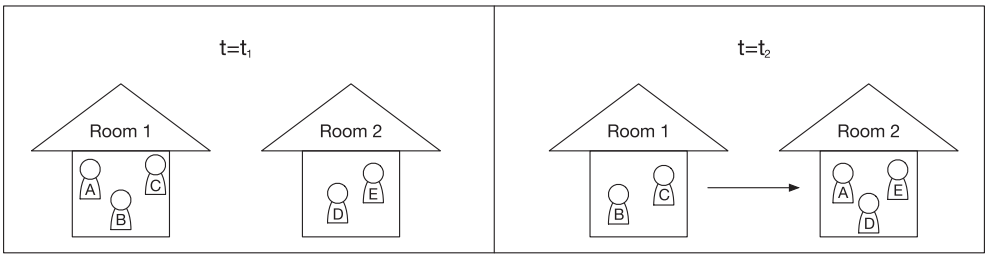


Fig. 2. The toy example of user A leaving room 1 and joining room 2. With the behavior of user A, the relationship between the two rooms will be updated.

Table 1
A list of notations used in this paper.

Notations	Description
$\bar{u}$ and $\bar{r}$	The static features of user u and room r respectively
$\boldsymbol{u}(t)$	The dynamic features of user u at time t
$\boldsymbol{r}^c(t)$	The cumulative features of room r at time t
$\boldsymbol{r}^i(t)$	The instantaneous features of room r at time t
$\boldsymbol{r}^e(t)$	The relationship features of room r at time t
$\boldsymbol{s}_u^r(t)$	The comprehensive features of room r at time t for user u
t^-	The timestamp right before time t
$\boldsymbol{U}_r^t$	The set of users who are in room r at time t
$enter(u,r,t)$	User u enters room r at time t
$leave(u,r,t)$	User u leaves room r at time t
$\hat{s}_u(t+\Delta t)$	The predicted comprehensive room features for user u at time $t+\Delta t$

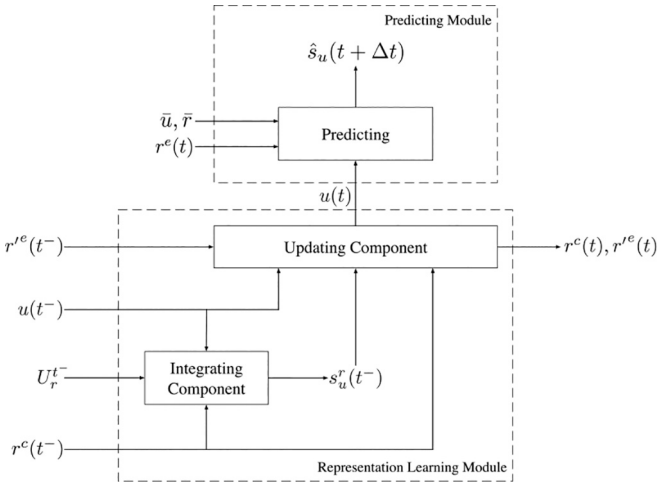


Fig. 3. The general framework of DRIVER.

learning and updating of vector representations. Within this module, both users' and rooms' dynamic representations are updated based on users' behavior paths of entering and leaving rooms. Upon user u's entrance into room r at time t, we will update the user's dynamic features $\boldsymbol{u}(t^-)$ with room r's comprehensive and user-specific features for user u, $\boldsymbol{s}_u^r(t^-)$. Such comprehensive features are obtained from the novel Integrating Component. Meanwhile, the room's cumulative features $\boldsymbol{r}^c(t^-)$ are also updated with $\boldsymbol{u}(t^-)$. Similar to the user-item co-updating method introduced in JODIE [9], these two update operations aim at tracking users' and rooms' characteristics with users' paths of entering rooms. Different from JODIE, which only considers the start of user-item interactions (i.e., room entrance), DRIVER also includes feature updates for the end of user-item interactions in users' behavior paths: when user u leaves room r' at time t^- and enters room r at time t, we will update $\boldsymbol{r}'^e(t^-)$, which captures the relationship features of room r', based on user u and room r. This allows the relationships between r' and other rooms to stay current in this dynamic environment. With the Integrating and Updating Components, we can get real-time and dynamic representations of users, rooms, and inter-room relationships.

The Predicting Module completes the prediction task to provide recommendations. It takes as inputs representations of users, rooms, as well as room-room relationships. Unlike many previous studies that predict a score for each item, DRIVER directly predicts the comprehensive features of the room that user u will enter at time $t+\Delta t$. We provide details about the Updating and Integrating Components and the Predicting Modules in subsequent subsections.

3.4. The updating component

To address users' dynamic behaviors in live streaming, three types of features need to be updated in our model: (1) users' dynamic features, (2) rooms' cumulative features, and (3) rooms' relationship features. The first two are updated upon users' entrance into rooms, while the last type is updated based on users' departure from rooms. Thus, the Updating Component in DRIVER represents a novel design that incorporates an instantaneous item graph into dynamic representation learning, and does so in a computationally efficient way, for dynamic user-item interactions that feature different types of behaviors in user path data (i.e., entrance and departure in live streaming).

First, $\boldsymbol{u}(t)$ represents user u's overall dynamic features. It is obtained based on her historical path of all the rooms she has ever entered till time t. The features will be updated each time this user enters a room. Because users' entrance into rooms form a natural temporal sequence, we choose the widely used RNN method for this update. When user u enters room r at time t, u's dynamic features are updated by room r's comprehensive features $\boldsymbol{s}_u^r(t^-)$. Formally, when $enter(u,r,t)$ occurs, $\boldsymbol{u}(t^-)$ is updated as specified in Eq. (1):

$$\boldsymbol{u}(t) = RNN_u(\boldsymbol{s}_u^r(t^-), \boldsymbol{u}(t^-)) \tag{1}$$

where $\boldsymbol{s}_u^r(t^-)$ is the latest comprehensive features of room r for user u before time t. This is obtained from the integrating component, which is described in Section 3.5.

Second, room r's cumulative features $\boldsymbol{r}^c(t)$ capture the long-term characteristics of the room such as the general content or style of shows till time t. Similar to $\boldsymbol{u}(t^-), \boldsymbol{r}^c(t^-)$ for room r is updated using RNN, each time user u enters room r at time t. Specifically, when the event $enter(u,r,t)$ occurs, $\boldsymbol{r}^c(t^-)$ is updated based on user u's dynamic features $\boldsymbol{u}(t^-)$, as described in Eq. (2):

$$\boldsymbol{r}^c(t) = RNN_r(\boldsymbol{u}(t^-), \boldsymbol{r}^c(t^-)) \tag{2}$$

To reduce the number of parameters to train, parameters of RNN_u and RNN_r are shared among all users and rooms respectively. Fig. 4a illustrates the procedure for updating $\boldsymbol{u}(t^-)$ and $\boldsymbol{r}^c(t^-)$.

Third, a room's relationship features capture its inter-room relationships with other rooms based on users' paths after leaving this room. When user u leaves room r' at time t^- and joins room r at time t, we update $\boldsymbol{r}'^e(t^-)$, the relationship features for room r', with two types of features: (1) $\boldsymbol{r}^c(t)$, the cumulative features of room r, so that we can capture characteristics of rooms that users choose after departing from this room; and (2) user u's dynamic features $\boldsymbol{u}(t)$. This design reflects

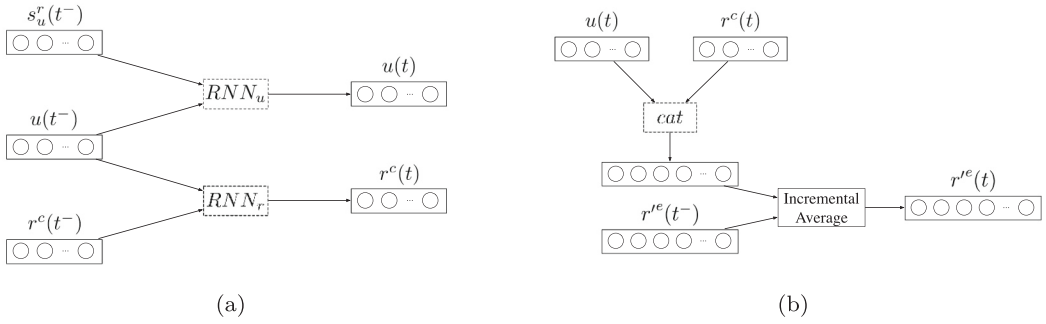


Fig. 4. The update process for $\boldsymbol{u}(\boldsymbol{t}^-)$ and $\boldsymbol{r}^{\boldsymbol{c}}(\boldsymbol{t}^-)$ (in panel a) and $\boldsymbol{r}'^{\boldsymbol{e}}(\boldsymbol{t}^-)$ (in panel b).

user heterogeneity as we discussed in the previous section: different users may make different choices after leaving the same room. For example, after watching a show about rock music, some users may want shows on hip-hop while other users may switch to shows on jazz. Overall, by including next room's features and user's features in this update procedure, $\boldsymbol{r}'^{\boldsymbol{e}}(\boldsymbol{t})$ can characterize the type of rooms different users have chosen after leaving room $\boldsymbol{r}'$.

One potential issue with this type of update is computational complexity. Note that the representations of users and rooms are continuously updated as soon as a user enters a room. This means the update of $\boldsymbol{r}'^{\boldsymbol{e}}(\boldsymbol{t}^-)$ must be done at the same pace to make real-time recommendations possible. However, most existing methods, such as CNN- or GNN-based models, rely on recursive and global computation on the whole item graph to capture the relationships among items after multi-epoch training on each new graph [37]. With the frequent changes to the room graph for live streaming, such an approach would incur high time complexity. To address this problem, we choose to update $\boldsymbol{r}'^{\boldsymbol{e}}(\boldsymbol{t}^-)$ with the average pooling operation, which can be calculated incrementally with high efficiency. Specifically, when user u is the (N + 1)-th user who leaves room r', and she subsequently enters room r, the update of $\boldsymbol{r}'^{\boldsymbol{e}}(\boldsymbol{t}^-)$ is defined as Eq. (3) and illustrated in Fig. 4b, where *cat* is the concatenation operation of vectors.

$$\boldsymbol{r}'^{\boldsymbol{e}}(\boldsymbol{t}) = (\boldsymbol{r}'^{\boldsymbol{e}}(\boldsymbol{t}^-) \times N + cat(\boldsymbol{u}(\boldsymbol{t}), \boldsymbol{r}^{\boldsymbol{c}}(\boldsymbol{t})))/(N+1) \tag{3}$$

3.5. The integrating component

Supported by the Integrated Framework for Consumer Path Modeling and social affordance theory, DRIVER integrates room's cumulative features and instantaneous features into a comprehensive and user-specific representation. In this way, the representation also explicitly captures social interactions within a room via instantaneous characteristics of room r's current audience, who could have great influence on the overall social atmosphere inside the room and even the style of ongoing shows. This subsection describes how the integrating component (1) obtains a room's instantaneous features $\boldsymbol{r}^{\boldsymbol{i}}(\boldsymbol{t}^-)$ by aggregating its current audience's/users' features and (2) integrates the cumulative and instantaneous features into the comprehensive features $\boldsymbol{s}_{\boldsymbol{u}}^{\boldsymbol{r}}(\boldsymbol{t}^-)$. Fig. 5 depicts the architecture of the integrating component.

As shown in Fig. 5, a room's instantaneous features are generated by aggregating dynamic features of users who are in the room. This is because individuals currently inside a room can serve as a good proxy for the social interactions and atmosphere of the room. Then, a room's instantaneous features are combined with its cumulative features to generate its comprehensive features. Specifically, to obtain instantaneous features $\boldsymbol{r}^{\boldsymbol{i}}(\boldsymbol{t}^-)$ for room r, we aggregate the dynamic features of each user $u' \in \boldsymbol{U}_{\boldsymbol{r}}^{\boldsymbol{t}^-}$ (the set of users who are currently in room r just before t), as defined in Eq. (4).

$$\boldsymbol{r}^{\boldsymbol{i}}(\boldsymbol{t}^-) = Agg_{u' \in \boldsymbol{U}_{\boldsymbol{r}}^{\boldsymbol{t}^-}}(\boldsymbol{u}'(\boldsymbol{t}^-)) \tag{4}$$

where *Agg* is the aggregation function and $\boldsymbol{u}'(\boldsymbol{t}^-)$ is learned in the updating component. Different aggregation operations can be used here.

With both cumulative features $\boldsymbol{r}^{\boldsymbol{c}}(\boldsymbol{t}^-)$ and instantaneous features $\boldsymbol{r}^{\boldsymbol{i}}(\boldsymbol{t}^-)$ for a room, we will aggregate them into room's comprehensive features specifically for user u. The reason we generate such user-specific room representations is again to address user heterogeneity—different users may have different levels of preferences over the same room's cumulative and instantaneous features: One user may favor the overall show content of a room (mainly reflected by its cumulative features from audience members' historical paths) while another user may value its current social atmosphere (mainly reflected by its instantaneous features from the current audience). Such preferences could also vary from one room to another. Therefore, to generate the comprehensive features of a room in a personalized way, we leverage the widely used attention mechanism [38] to capture users' attention levels to a room's cumulative and instantaneous features. Formally, we get a room's comprehensive user-specific features with the following equations:

$$o_{ur}^c = \sigma(\boldsymbol{w}_c \cdot cat(\boldsymbol{u}(\boldsymbol{t}^-), \boldsymbol{r}^{\boldsymbol{c}}(\boldsymbol{t}^-)) + b_1) \tag{5}$$

$$o_{ur}^i = \sigma(\boldsymbol{w}_i \cdot cat(\boldsymbol{u}(\boldsymbol{t}^-), \boldsymbol{r}^{\boldsymbol{i}}(\boldsymbol{t}^-)) + b_2) \tag{6}$$

$$a_{ur}^c = \exp(o_{ur}^c) / (\exp(o_{ur}^c) + \exp(o_{ur}^i)) \tag{7}$$

$$a_{ur}^i = 1 - a_{ur}^c \tag{8}$$

$$\boldsymbol{s}_{\boldsymbol{u}}^{\boldsymbol{r}}(\boldsymbol{t}^-) = a_{ur}^c \times \boldsymbol{r}^{\boldsymbol{c}}(\boldsymbol{t}^-) + a_{ur}^i \times \boldsymbol{r}^{\boldsymbol{i}}(\boldsymbol{t}^-) \tag{9}$$

Eqs. (5)–(8) get the relative weights of a room's cumulative and instantaneous features and Eq. (9) combines these features according to these weights to generate room r's comprehensive user-specific features specifically for user u. $\boldsymbol{w}_c, \boldsymbol{w}_i \in \boldsymbol{R}^{2\times d}, b_1, b_2 \in \boldsymbol{R}$ are all learnable parameters, and σ is the sigmoid function.

3.6. The predicting module

With all the representations learned above, when user u leaves room r at time t, DRIVER first predicts the comprehensive features of the room she will prefer right before she chooses a room at time $t + \Delta t$. Here we also adopt the projection operation proposed in JODIE [9] to model the potential temporal drift of user's characteristics (i.e., the change of users' dynamic features over time). The projection operation first converts time difference into a time-context vector and then scales the user's dynamic features $\boldsymbol{u}(\boldsymbol{t})$:

$$\widehat{\boldsymbol{u}}(\boldsymbol{t} + \Delta \boldsymbol{t}) = (\boldsymbol{1} + (\boldsymbol{w}_t \times \Delta t)) \odot \boldsymbol{u}(\boldsymbol{t}) \tag{10}$$

where the learnable parameter $\boldsymbol{w}_t \in \boldsymbol{R}^d$ is used to convert time difference into the time-context vector, $\boldsymbol{1}$ is a vector whose entries are all 1 and $\odot$ is the element-wise product.

For user u, the Predicting Module takes three types of features as inputs. Naturally, the first type is about the user, including $\widehat{\boldsymbol{u}}(\boldsymbol{t} + \Delta \boldsymbol{t})$, the

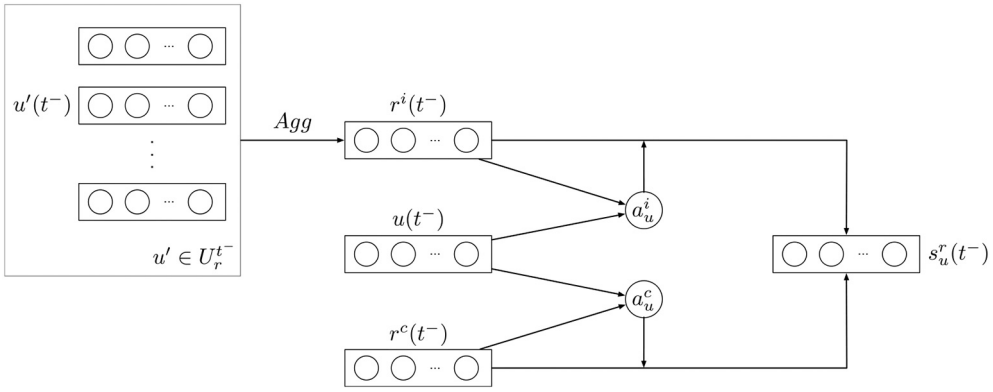


Fig. 5. The Integrating Component.

user's dynamic features projected at the prediction time, and her static feature $\bar{\boldsymbol{u}}$. The second type is about the room r that u just left. Because dynamic features of the room are already incorporated into u's dynamic representations when she enters the room (via the Updating Component described in Section 3.4 and illustrated in Fig. 4a), we only include the room's static feature $\bar{\boldsymbol{r}}$ as a direct input to the prediction. The last type is about relationships of rooms based on users' sequential behaviors. If user u just leaves room r, then $\boldsymbol{r}^e(t)$, which captures r's relationships with other rooms, is also fed into the prediction. Formally, when user u leaves room r at time t, the Predicting Module takes $\widehat{\boldsymbol{u}}(t+\Delta t), \bar{\boldsymbol{u}}, \bar{\boldsymbol{r}}$ and $\boldsymbol{r}^e(t)$ as inputs and predicts $\widehat{\boldsymbol{s}}_u(t+\Delta t)$ as defined in Eq. (11), where $\boldsymbol{W}_s$ is a matrix to be learned.

$$\widehat{\boldsymbol{s}}_u(t+\Delta t) = \boldsymbol{W}_s \cdot cat(\widehat{\boldsymbol{u}}(t+\Delta t), \bar{\boldsymbol{u}}, \boldsymbol{r}^e(t), \bar{\boldsymbol{r}}) \tag{11}$$

To train this predictive model, we define the *Mean Square Error* between the predicted comprehensive features for user u and the comprehensive features of the room that u actually entered as the loss function (see Eq. (12)). We also add the magnitudes of feature updates for users and rooms into Eq. (12) to ensure that these dynamic features are updated smoothly. λ_u and λ_r are hyper-parameters. The minimization problem can be solved with gradient descent.

$$\begin{aligned} L = & \sum_{(u,r,t)} \|\widehat{\boldsymbol{s}}_u\left(t\right) - \boldsymbol{s}_u^r\left(t\right)\|_2^2 \\ & + \lambda_u \|\boldsymbol{u}\left(t\right) - \boldsymbol{u}\left(t^-\right)\|_2^2 + \lambda_r \|\boldsymbol{r}^c\left(t\right) - \boldsymbol{r}^c\left(t^-\right)\| \end{aligned} \tag{12}$$

The prediction outcome enables us to calculate the Euclidean distance between actual representations to each room and the prediction. Rooms are then ranked by such distance in an ascending order (i.e., in the descending order of similarity with the prediction). Top K rooms from the ranked list will be recommended to user u as candidates for her next room.

4. Experiments and results

We conduct experiments on real-world datasets from two different types of online streaming platforms and compare the performance of the proposed DRIVER model with other state-of-the-art recommendation models. Then, we use ablation studies to demonstrate that each new component of DRIVER contributes to its performance. Sensitivity analyses also show the robustness of DRIVER's performance.

4.1. Datasets

The first dataset, **Live-Singing**, is from a popular live streaming platform dedicated to online singing, with services in North America and South Asia.[3] Hosts can start live-singing rooms, and other users can enter to watch others' performance or even sing a song if they want to. The dataset contains 26,604 users' interactions with 936 rooms between June 3rd and June 23th, 2019. The second dataset, **HuaJiao**, is from a large-scale live streaming platform in China.[4] Compared to Live-Singing, HuaJiao features more diverse shows of singing, dancing, gaming, etc. The dataset contains 20,000 users' interactions with 1,000 rooms from Feb 1st to Feb 28th,2018. Summary statistics of the two datasets are listed in Table 2, where an "interaction" between a user and a room refers to the user's entrance into or departure from the room. Note that the two datasets were used separately to train two different DRIVER models, one for each platform.

4.2. Benchmark methods and experimental settings

We choose a variety of recommendation methods as benchmark methods. All of the seven benchmark methods are intorduced below, including one classic matrix factorization model, two static representation learning models based on deep learning, three sequential recommendation models, and one dynamic representation learning method.

- **MF-BPR** [7] is a classic matrix factorization recommendation algorithm that learns latent vectors for both users and items with the BPR criterion.
- **NGCF** [39] is a static representation learning model which leverages multi-layer GNN to learn representations from user-item interaction graph with the BPR loss.

Table 2
Basic statistics of datasets.

	#users	#rooms	#interactions	density	#interactions per day
Live-Singing	26,604	936	921,332	0.0370	46,067
HuaJiao	20,000	1,000	1,726,592	0.0863	61,664

[3] Due to a non-disclosure agreement, we cannot disclose the name of the platform.
[4] https://www.huajiao.com.

- **LightGCN** [8] is an extension of NGCF. It drops the non-linear transformations when conducting GNN and claims better performance.
- **SR-GNN** [21] is a GNN-based sequential recommendation model that captures the relationships among items (i.e., rooms in our case) in the same session.
- **TiSASRec** [40] is a sequential recommendation model based on self-attention. Both the sequential patterns and time intervals between every two items in interaction sequences are considered in this model.
- **SSE-PT** [41] is another sequential recommendation model which captures user-specific sequential patterns using self-attention.
- **JODIE** [9] is a state-of-the-art dynamic presentation learning model for sequential recommendations. It jointly learns and updates user and item representations upon user-item interactions.

All the models are trained 30 times with different random seeds to get a stable result, and the average performance is shown. We use the commonly used metrics MRR, NDCG, and Recall to evaluate the recommendation performance. Details about their calculations are in Appendix B. To ensure reproducibility, we follow guidelines in [42–44] and provide details of our model and experiments in Appendix B and C. The code and a simulated dataset are available at https://github.com/GarrettGaoge/DRIVER.

4.3. Results

Table 3 summarizes the average performance of all models on the test set. The best performer for each measure is shown in bold. The last row shows the relative improvement of DRIVER compared to the best benchmark model and whether the improvement is statistically significant (with paired t-tests and *** indicates p-value < 0.001).

As Table 3 shows, the proposed DRIVER model dominates the benchmark methods on all performance measures for both datasets. Among benchmark methods, those that learn only static representations for users and rooms, such as MF-BPR, NGCF, and LightGCN, generally have lower performance. Sequential recommenders, such as SR-GNN, TiSASRec, and SSE-PT, have better performance because they consider sequential patterns based on users' paths of entering and leaving rooms. However, although session information or self-attention are used in these methods to capture time sequences, they still rely on static representations.

JODIE is the state-of-the-art dynamic representation learning model that continuously updates the representations of users and rooms when users enter rooms. Its good performance among benchmark methods highlights the benefits of learning dynamic representations when user-item/room interactions occur at a very high pace, as in live streaming platforms. However, JODIE does not consider users' paths after leaving a room. Moreover, JODIE treats all historical paths of users equally and does not capture social interactions among current audience members within a room. Thus, DRIVER's better performance compared to JODIE demonstrates the value of considering two unique characteristics of live streaming: dynamic inter-room relationships and the importance of social interactions in a room. Overall, DRIVER's better performance than benchmark methods demonstrates that our approach to capturing users' behavior paths of entering, staying in, and leaving rooms are effective.

Moreover, we also evaluate DRIVER's performance for diverse types of shows in HuaJiao. The results show that DRIVER's better performance over JODIE is consistent across all the 8 types of shows in the HuaJiao dataset–the improvement in MRR ranges from 3.81% (for show type *Fun*) to 11.16% (for show type *Knowledge Sharing*). Appendix D shows more details about the heterogeneity of the two live streaming platforms and the datasets used.

4.4. Ablation studies

As we mentioned earlier in this paper, DRIVER incorporates two new components to capture unique characteristics of live streaming: dynamic inter-room relationships and instantaneous features of rooms. To evaluate how each of the two new components contributes to DRIVER's performance, we conduct ablation studies for two variations of DRIVER. The first variation (DRIVER1) removes rooms' relationship features, i.e., $\boldsymbol{r}^e(t)$, and substitutes them with comprehensive features of the last room that a user visited when doing prediction in Eq. (11). The second variation (DRIVER2) removes rooms' instantaneous features based on the current audience and user-specific comprehensive features. Instead, rooms are only represented by cumulative features as in JODIE. Table 4 compares the performance of the two variations with that of the full DRIVER model, along with two strong benchmark methods, TiSASRec and JODIE.

According to Table 4, the removal of each component leads to deteriorating performance compared to the full DRIVER model. At the same time, both variations still outperform the two strong benchmark methods. These results indicate that each of the two new components makes meaningful contributions in improving DRIVER's performance over benchmark methods. (*** means the result is different from

Table 3
Performance comparison.

		MRR	Recall@5	Recall@10	NDCG@5	NDCG@10
Live-Singing	MF-BPR	0.2995	0.4372	0.5746	0.3901	0.4340
	NGCF	0.3033	0.4487	0.5776	0.3954	0.4373
	LightGCN	0.3964	0.4815	0.5884	0.4420	0.4733
	SR-GNN	0.4595	0.5383	0.6332	0.4617	0.4925
	TiSASRec	0.5067	0.6371	0.7294	0.5227	0.5527
	SSE-PT	0.4581	0.6217	0.7258	0.4807	0.5144
	JODIE	0.5130	0.6652	0.7366	0.5395	0.5627
	DRIVER	**0.5488**	**0.6989**	**0.7896**	**0.5721**	**0.6015**
		(±0.0023)	(±0.0017)	(±0.0016)	(±0.0021)	(±0.0020)
	Improvement	6.97%***	5.06%***	7.19%***	6.05%***	6.90%***
HuaJiao	MF-BPR	0.2138	0.3149	0.4169	0.3297	0.3579
	NGCF	0.2134	0.3150	0.4145	0.3296	0.3566
	LightGCN	0.3847	0.3332	0.4138	0.3824	0.3948
	SR-GNN	0.4237	0.4696	0.5120	0.4240	0.4376
	TiSASRec	0.4574	0.5709	0.6391	0.4719	0.4940
	SSE-PT	0.4147	0.5086	0.5889	0.4211	0.4470
	JODIE	0.4568	0.5769	0.6396	0.4751	0.4955
	DRIVER	**0.4982**	**0.6006**	**0.6513**	**0.5138**	**0.5303**
		(±0.0010)	(±0.0025)	(±0.0045)	(±0.0012)	(±0.0018)
	Improvement	8.93%***	4.11%***	1.83%***	8.15%***	7.01%***

Table 4
Ablation study results.

		MRR	Recall@5	Recall@10	NDCG@5	NDCG@10
Live-Singing	TiSASRec	0.5067	0.6371	0.7294	0.5227	0.5527
	JODIE	0.5130	0.6652	0.7366	0.5395	0.5627
	DRIVER1	0.5205***	0.6896***	0.7896***	0.5471***	0.5796***
	DRIVER2	0.5460***	0.7006**	0.7878	0.5702***	0.5984***
	DRIVER	0.5488	0.6989	0.7896	0.5721	0.6015
HuaJiao	TiSASRec	0.4574	0.5709	0.6391	0.4719	0.4940
	JODIE	0.4568	0.5769	0.6396	0.4751	0.4955
	DRIVER1	0.4981	0.5981***	0.6409***	0.5143	0.5282***
	DRIVER2	0.4863***	0.5890***	0.6387***	0.5022***	0.5183***
	DRIVER	0.4982	0.6006	0.6513	0.5138	0.5303

DRIVER according to paired t-tests with p-value < 0.001, ** means p-value < 0.01)

4.5. The robustness of performance

An important hyper-parameter for DRIVER is the dimension of vector representations. Our experiments above use 128 as the dimension. To check if DRIVER offers robust performance when different dimensions are used, we vary the dimensions and compare DRIVER's performance on both datasets in Fig. 6. We can see that the performance is not sensitive to the change of representation dimensions.

Besides this hyper-parameter, we also evaluate whether DRIVER's performance is robust to different amount of training data. For both datasets, we train DRIVER based on various length of training periods—2 days to 20 days for Live-Singing and 6 days to 26 days for HuaJiao—and use the remaining data for testing. Fig. 7 shows the performance of DRIVER compared to the benchmark methods on dataset Live-Singing and we get similar results in HuaJiao (corresponding figures are shown in Appendix E).

The sensitivity analyses reveal two interesting findings. *First*, DRIVER outperforms all other models with different amounts of data on both datasets. In other words, DRIVER's performance improvement over other methods is robust against changes in training data size—while having more training data is generally beneficial, having less training data does not cause great drops in DRIVER's performance. *Second*, DRIVER needs much less training data to achieve the same level of performance. For example, on the Live-Singing dataset, DRIVER trained with only 2 days of data performs better than other models trained with 20 days. On the HuaJiao dataset, to get comparable MRR with DRIVER trained with 6 days of data, benchmark methods need 20 or more days of data. This highlights the value of the additional signals DRIVER leverages from user path data to make recommendations.

Overall, the sensitivity analyses demonstrate the robustness of DRIVER's performance to hyper-parameter values and training data sizes. Such properties are highly desirable for the real-world adoption of a recommender system.

5. Discussions

While interpreting deep learning models and vector representations learned from such models remains a challenge, we dig deeper into our model to see what it learns. This helps us better understand if the model learns what we intend for it to learn, ensure that the performance gain from DRIVER is not accidental, and validate our model design guided by empirical evidence and theory. Using the Live-Singing dataset, our investigations focus on the two novel components of DRIVER: the incorporation of the current audience into the learning of room representations and the dynamic representations of inter-room relationships.

First, when generating comprehensive features for a room, DRIVER aggregates the room's cumulative features and instantaneous features with the attention mechanism (Eq. (9)). The attention learned for a user would indicate how much this user values a room's overall characteristics (represented by a^c_{ur}) versus its instantaneous atmosphere (represented by a^i_{ur}). Thus, we calculate users' attention on each room's instantaneous features (a^i_{ur}) when they choose rooms in the testing period and analyze attention values from two perspectives. From the perspective of users, we calculate each user's averaged attention $\bar{a}^i_u$ on room's instantaneous features $\bar{a}^i_u = Mean_{r\in \boldsymbol{M}(u)}(a^i_{ur})$, where $\boldsymbol{M}(u)$ is the set of all the rooms user u has entered and *Mean* is the function to get the mean value. Larger $\bar{a}^i_u$ means user u usually cares more about rooms' social interactions and atmosphere (i.e., reflected by its current audience), and vice versa. $\bar{a}^i_u(\forall u \in \boldsymbol{U})$ has a mean of 0.53 and a standard deviation of 0.01.

Similarly, from the perspective of rooms, we average the attention to rooms' instantaneous features across users $\bar{a}^i_r = Mean_{u\in \boldsymbol{U}(r)}(a^i_{ur})$, where $\boldsymbol{U}(r)$ is the set of all the users who have entered room r. The mean and standard deviation of $\bar{a}^i_r(\forall r \in \boldsymbol{M})$ are 0.52 and 0.01 respectively. Because the relative weights of a room's cumulative and instantaneous features a^c_{ur} and a^i_{ur} sum up to 1, results from both perspectives show that most users do take the current audience in a room as an important factor, sometimes even slightly more important than the room's long-term

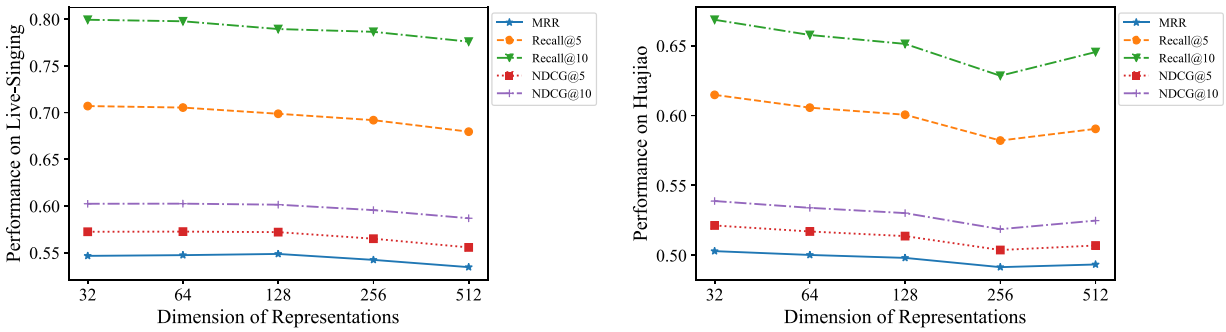


Fig. 6. The performance of DRIVER with different dimension of vector representations on two datasets—Live-Singing (left) and HuaJiao (right).

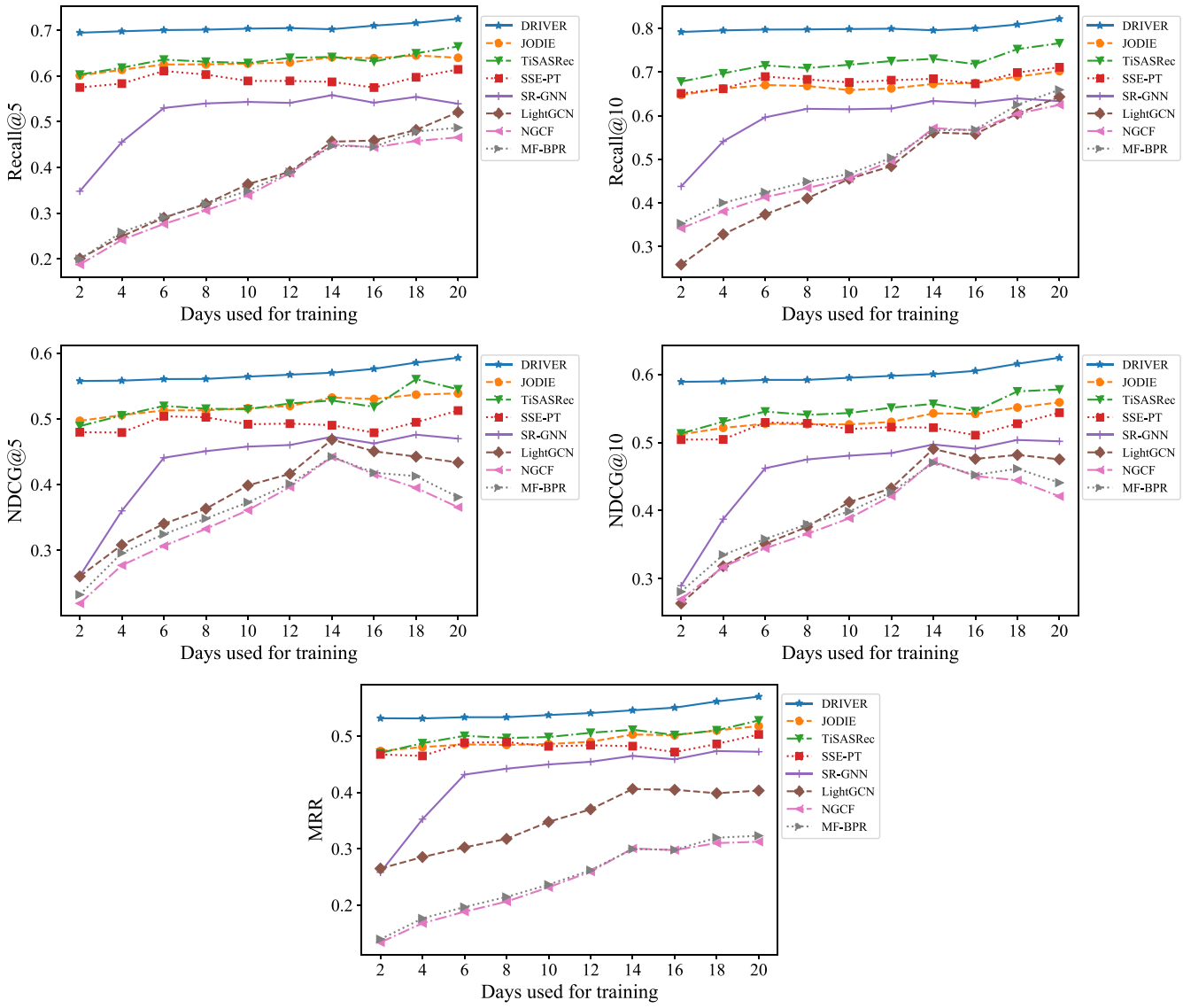


Fig. 7. Performance of different models with different training sets on Live-Singing.

characteristics, when choosing which room to enter. In other words, as we have shown in our ablation analyses, it is indeed valuable to capture characteristics of a room's current audience as a proxy for its social interactions when learning the room's representations.

As we discussed above, live streaming features high dynamics in user-room interactions and our model is designed to capture such dynamics. Therefore, our second investigation into the model explores if representations learned by DIRVER are indeed dynamic, including rooms' cumulative features $\boldsymbol{r}^c$, instantaneous features $\boldsymbol{r}^i$, and inter-room relationship features $\boldsymbol{r}^e$. To measure the dynamics in a room's features, we first calculate cosine similarities between all pairs of rooms' features at time t_b, the beginning of the testing period. For example, Eq. (13) calculates $sim^c_{r_1,r_2}(t_b)$, the similarity between room r_1's and r_2's cumulative features at t_b, where the superscript T means the transpose of a vector and $\|\cdot\|_2$ is the second-norm. The similarity between their instantaneous features $sim^i_{r_1,r_2}(t_b)$ and the similarity between their relationship features $sim^e_{r_1,r_2}(t_b)$ are calculated in a similar way. Then for each room r, we can find three sets of top K similar rooms—$TK^c_r(t_b), TK^i_r(t_b)$ and $TK^e_r(t_b)$, based on similarities on cumulative, instantaneous and relationship features at t_b respectively.

$$sim^c_{r_1,r_2}(t_b) = \frac{\boldsymbol{r}^c_1(t_b)^T \cdot \boldsymbol{r}^c_2(t_b)}{\|\boldsymbol{r}^c_1(t_b)\|_2 \times \|\boldsymbol{r}^c_2(t_b)\|_2} \tag{13}$$

Following the same procedure, we can find inter-room similarities between rooms at time t_e, the end of the test period, and identify room r's three sets of top K similar rooms—$TK^c_r(t_e), TK^i_r(t_e)$ and $TK^e_r(t_e)$, based on similarities on cumulative, instantaneous and relationship features at t_e respectively.

Then by comparing a room's top K similar rooms at the beginning and end of the testing period, we can reveal the dynamics in rooms' representations over the 7-day period. Take the comparison of top K similar rooms based on cumulative features as an example. For room r, we calculate the Jaccard Coefficient JC^c_r between $TK^c_r(t_b)$ and $TK^c_r(t_e)$. A higher Jaccard Coefficient means a room's top K similar rooms based on cumulative features do not change much. Jaccard Coefficients for instantaneous features (JC^i_r) and relationship features (JC^e_r) can be calculated in a similar way. Table 5 shows the means and standard deviations for Jaccard Coefficients with $K = 5$ and $K = 10$, while Fig. 8 draws the distributions of Jaccard Coefficients over rooms.

For the three types of features and two K values, the mean Jaccard Coefficients are all below 0.15. Also, more than 600 out of 936 rooms

Table 5
Means and standard deviations of Jaccard Coefficients for top K similar rooms at the beginning and end of the testing period of Live-Singing.

	$K = 5$		$K = 10$	
	Mean	Standard Deviation	Mean	Standard Deviation
Cumulative Features	0.12	0.22	0.13	0.21
Instantaneous Features	0.08	0.14	0.09	0.13
Relationship Features	0.12	0.26	0.11	0.20

have Jaccard Coefficients lower than 0.1. In other words, a room's top K similar rooms experience radical changes during the 7-day period. These results demonstrate that representations learned by DRIVER are indeed very dynamic over time, which reflects one of the key characteristics of live streaming rooms. Moreover, as one would expect, instantaneous features have lower similarities than cumulative and relationship features. This is because instantaneous features are based directly on a room's current audience, which is the most dynamic aspect of a room.

6. Conclusions and future work

This paper proposes DRIVER, a novel dynamic representation learning model for recommending rooms in live streaming platforms. Guided by the Integrated Framework for Consumer Path Modeling and social affordance theory, the proposed model addresses two unique and important features of live streaming—the value of the current audience within a room and the dynamic inter-room relationships when users switch rooms frequently. The model designs two novel components to learn and update dynamic representations for both users and rooms based on users' behavior paths of entering, staying in, and leaving rooms. Three types of dynamic features of a room are learned: cumulative features based on users' entrance into rooms in the past, instantaneous features based on the audience members who are currently in a room, and relationship features reflected in users' subsequent behavior paths after leaving a room. With these representations, we leverage the features of users and rooms as well as inter-room relationships to predict which room a user will choose next.

Through extensive experiments on datasets from two real-world live streaming platforms, we demonstrate the improved performance of DRIVER and summarize the main findings as follows:

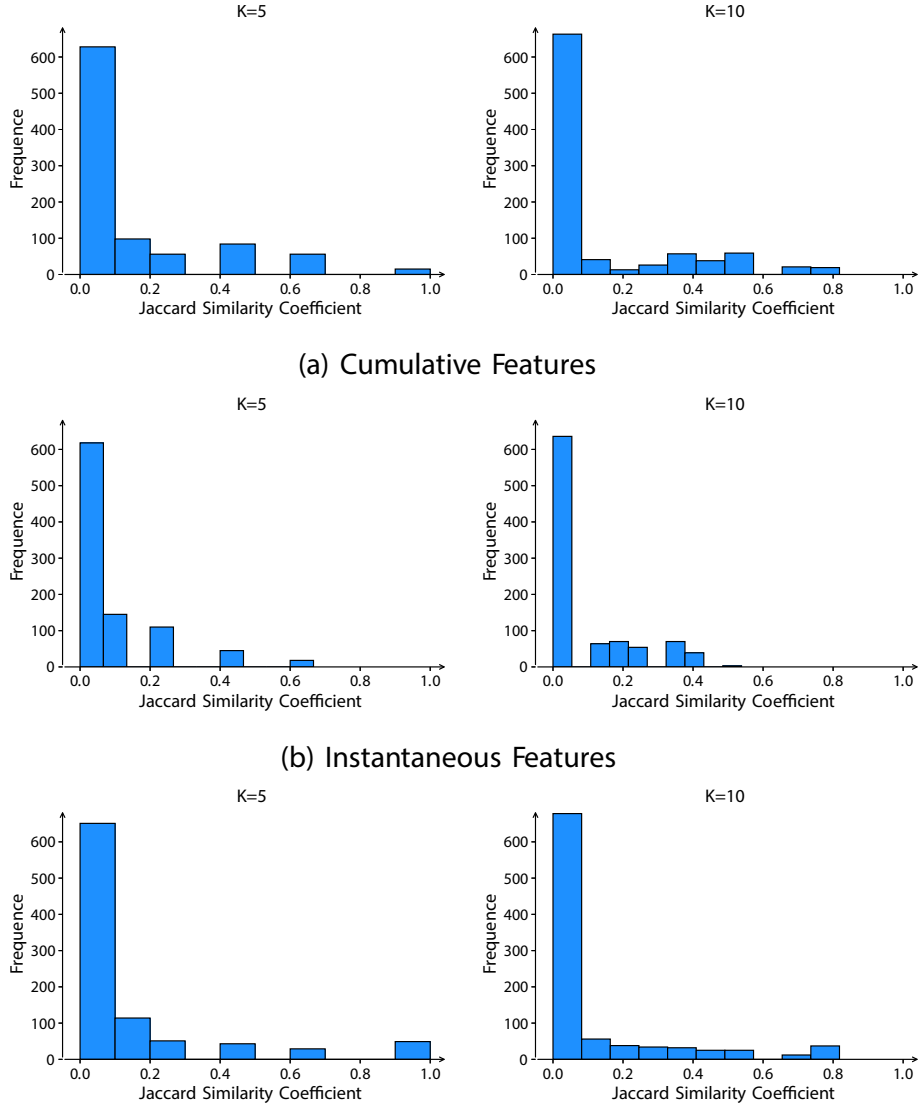


Fig. 8. Distributions of Jaccard Coefficients for top K similar rooms at the beginning and end of the testing period of Live-Singing.

First, in live streaming, both hosts and audience enjoy the freedom of choosing what they perform and watch respectively, leading to ever-changing relationships among rooms. The incorporation of such dynamic inter-room relationships based on the Consumer Path Modeling helps DRIVER to improve its performance in room recommendation tasks.

Second, live streaming platforms enable various types of social interactions between the audience or between a host and the audience. Inspired by the social affordance theory, DRIVER's approach to representing the social atmosphere inside a room with characteristics of its current audience can also help to generate more accurate room recommendations.

These findings have important implications for the design of recommendation systems in live streaming platforms and beyond. *First*, with highly dynamic host and audience behaviors in live streaming platforms, room recommendation systems should take into consideration the dynamic properties of users and rooms, as well as the social atmosphere inside rooms. Designers of room recommendation systems can adopt DRIVER's framework, yet train and tweak the model with their own data. In addition, because DRIVER is designed as a generalizable model without considering platform-specific features, different platforms can also extend DRIVER to incorporate unique characteristics of their services to further improve its performance. Such improvements in room recommendations can help live streaming platforms reduce users' search costs and offer a greater entertainment experience, allowing users and hosts to get more engaged. This can result in financial and non-monetary benefits for both platforms and their customers.

Second, methodologically, the implications of the DRIVER model are not limited to live streaming recommendations only. DRIVER's performance for live streaming recommendations highlights that learning dynamic item representations and directly aggregating users' features are feasible ways to model dynamic user-item and users' social interactions in cyberspace. We believe DRIVER can also be extended to other contexts where real-time social interactions among users are important for item properties or user-item interactions occur at a fast pace.

There are also several interesting directions for further work. For example, many live streaming platforms allow users to follow each other and form an explicit social network, which could provide valuable information to better understand user behaviors and recommend rooms. In addition, this study uses characteristics of the current audience as a proxy for social interactions inside a room, as this enables generalization to different live streaming platforms and rooms with different types of shows. When making room recommendations for a specific live streaming platform or a specific type of show, investigating fine-grained behaviors of the audience members inside a room, such as the nature, volume, and content of their interactions with each other and the host (e.g., virtual gifting, meme, and sentiment of comments) should also help to better capture the social atmosphere inside a room. Similarly, adding more static features for each room (e.g., the text description of each room) and dynamic features related to the audio and video content of ongoing shows should also improve the performance of room recommendations and help to address the cold-start problem. However, the extraction of these features and their contributions to recommendation performance may depend heavily on the specifics of the live streaming platform and the types of shows being hosted in rooms. We also noticed that all the recommendation models we evaluated, including DRIVER, work better for Live-Singing than for HuaJiao. We conjecture that the performance difference is due to HuaJiao's higher diversity in the types of shows, hosts, and audience, making it a more challenging platform to use one model for all room recommendations. It would be interesting to analyze factors that lead to lower performance by so many different recommendation models, but this would require a systematic investigation that goes beyond just two live streaming platforms. Another direction for future research would be to train and evaluate our model with more diverse datasets from other types of live streaming platforms, so that we can better validate its performance for the general task of room recommendations for live streaming.

CRediT authorship contribution statement

Ge Gao: Methodology, Software, Validation, Investigation, Writing-original-draft, Visualization. **Hongyan Liu:** Conceptualization, Methodology, Resources, Writing-review-editing, Supervision, Project-administration, Funding-acquisition. **Kang Zhao:** Conceptualization, Methodology, Writing-review-editing, Supervision, Project-administration.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We promised to provide our codes on GitHub with a synthetic dataset after this paper is accepted.

Acknowledgements

This work was supported in part by the National Social Science Major Program with Grant No. 20&ZD161. Due to space limitation, we move appendix to the online Supplementary materials.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, athttps://doi.org/10.1016/j.dss.2023.113957.

References

[1] Z. Lu, H. Xia, S. Heo, D. Wigdor, You watch, you give, and you engage: A study of live streaming practices in china, in: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 1–13.
[2] Kaja J. Fietkiewicz, Katrin Scheibe, Information behavior on social live streaming services****, J. Inf. Sci. Theory Pract. 4 (2016) 6–20.
[3] Z. Li, X. Fang, X. Bai, O.R.L. Sheng, Utility-Based Link Recommendation for Online Social Networks, Manage. Sci. 63 (2016) 1938–1952.
[4] R. Mishra, P. Kumar, B. Bhasker, A web recommendation system considering sequential information, Decis. Support Syst. 75 (2015) 1–10.
[5] M. Zihayat, A. Ayanso, X. Zhao, H. Davoudi, A. An, A utility-based news recommendation system, Decis. Support Syst. 117 (2019) 14–27.
[6] E. Yu, C. Jung, H. Kim, J. Jung, Impact of viewer engagement on gift-giving in live video streaming, Telematics Inform. 35 (2018) 1450–1460.
[7] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, Bpr: Bayesian personalized ranking from implicit feedback, in: Uncertainty in Artificial Intelligence, Mchine Lerning Lb, Universiy of Hideshei, Montreal, Quebec, Canada, 2009, pp. 452–461.
[8] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, M. Wang, Lightgcn: Simplifying and powering graph convolution network for recommendation, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 639–648.
[9] S. Kumar, X. Zhang, J. Leskovec, Predicting dynamic embedding trajectory in temporal interaction networks, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 1269–1278.
[10] S.K. Hui, P.S. Fader, E.T. Bradlow, Path Data in Marketing: An Integrative Framework and Prospectus for Model Building, Mark. Sci. 28 (2008) 320–335. Publisher: INFORMS.
[11] S. O'Riordan, J. Feller, T. Nagle, A categorisation framework for a feature-level analysis of social network sites, J. Decis. Syst. 25 (2016) 244–262.
[12] N. Chaudhuri, G. Gupta, V. Vamsi, I. Bose, On the platform but will they buy? Predicting customers' purchase behavior using deep learning, Decis. Support Syst. 149 (2021), 113622.
[13] Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (2013) 1798–1828.
[14] W. Wen, D. Zeng, J. Bai, K. Zhao, Z. Li, Learning embeddings based on global structural similarity in heterogeneous networks, IEEE Intell. Syst. (2020), 1–1.

[15] Y. Song, N. Sahoo, E. Ofek, When and How to Diversify—A Multicategory Utility Model for Personalized Content Recommendation, Manage. Sci. 65 (2019) 3737–3757.
[16] Y. Wang, N. Du, R. Trivedi, L. Song, Coevolutionary latent feature processes for continuous-time user-item interactions, in: Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16, Curran Associates Inc., Red Hook, NY, USA, 2016, pp. 4554–4562.
[17] E. Liebman, M. Saar-Tsechansky, P. Stone, The Right Music at the Right Time: Adaptive Personalized Playlists Based on Sequence Modeling, MIS Q. 43 (2019) 765–786.
[18] S. Rendle, C. Freudenthaler, L. Schmidt-Thieme, Factorizing personalized markov chains for next-basket recommendation, in: Proceedings of the 19th International Conference on World Wide Web, WWW '10, Association for Computing Machinery, New York, NY, USA, 2010, pp. 811–820.
[19] J. Tang, K. Wang, Personalized top-n sequential recommendation via convolutional sequence embedding, in: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 565–573.
[20] W.L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, pp. 1025–1035.
[21] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, T. Tan, Session-based recommendation with graph neural networks, in: The Thirty-Third AAAI Conference on Artificial Intelligence, vol. 33, AAAI Press, 2019, pp. 346–353.
[22] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, D. Yin, Graph neural networks for social recommendation, in: The World Wide Web Conference, WWW '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 417–426.
[23] B. Perozzi, R. Al-Rfou, S. Skiena, DeepWalk: Online Learning of Social Representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, ACM, New York, NY, USA, 2014, pp. 701–710.
[24] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16, Curran Associates Inc., Red Hook, NY, USA, 2016, pp. 3844–3852.
[25] W.L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, Curran Associates Inc., Long Beach, California, USA, 2017, pp. 1025–1035.
[26] A.L. Montgomery, S. Li, K. Srinivasan, J.C. Liechty, Modeling Online Browsing and Path Analysis Using Clickstream Data, Mark. Sci. 23 (2004) 579–595. Publisher: INFORMS.
[27] J. Li, A. Abbasi, A. Cheema, L.B. Abraham, Path to Purpose? How Online Customer Journeys Differ for Hedonic Versus Utilitarian Purchases, J. Mark. 84 (2020) 127–146.
[28] A.P. Vrechopoulos, R.M. O'Keefe, G.I. Doukidis, G.J. Siomkos, Virtual store layout: an experimental comparison in the context of grocery retail, J. Retail. 80 (2004) 13–22.
[29] J.J. Argo, D.W. Dahl, R.V. Manchanda, The Influence of a Mere Social Presence in a Retail Context, J. Consum. Res. 32 (2005) 207–212.
[30] Y. Deng, J.J. Hou, X. Ma, S. Cai, A dual model of entertainment-based and community-based mechanisms to explore continued participation in online entertainment communities, Cyberpsychol. Behav. Soc. Netw. 16 (2013) 378–384.
[31] L. Suárez, C. Thio, S. Singh, Attachment styles, motivations, and problematic use of massively multiplayer online games, Int. Proc. Econ. Dev. Res. 53 (2012) 45–49.
[32] L. Gu, A.L. Jia, Player activity and popularity in online social games and their implications for player retention, in: 2018 16th Annual Workshop on Network and Systems Support for Games (NetGames), 2018, pp. 1–6.
[33] M. Sjöblom, M. Törhönen, J. Hamari, J. Macey, The ingredients of Twitch streaming: Affordances of game streams, Comput. Hum. Behav. 92 (2019) 20–28.
[34] Z. Hilvert-Bruce, J.T. Neill, M. Sjöblom, J. Hamari, Social motivations of live-streaming viewer engagement on Twitch, Comput. Hum. Behav. 84 (2018) 58–67.
[35] K. Zhao, Y. Hu, K. Hong, J. Westland, Understanding Characteristics of Popular Streamers on Live Streaming Platforms: Evidence from Twitch.tv, J. Assoc. Inf. Syst. (2020).
[36] G.D. Harrell, M.D. Hutt, J.C. Anderson, Path analysis of buyer behavior under conditions of crowding, J. Mark. Res. 17 (1980) 45–51.
[37] J. Li, H. Dani, X. Hu, J. Tang, Y. Chang, H. Liu, Attributed network embedding for learning in a dynamic environment, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 387–396.
[38] S. Chaudhari, V. Mithal, G. Polatkan, R. Ramanath, An attentive survey of attention models, ACM Trans. Intell. Syst. Technol. 12 (2021).
[39] X. Wang, X. He, M. Wang, F. Feng, T.-S. Chua, Neural graph collaborative filtering, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 165–174.
[40] J. Li, Y. Wang, J. McAuley, Time interval aware self-attention for sequential recommendation, in: Proceedings of the 13th International Conference on Web Search and Data Mining, WSDM '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 322–330.
[41] L. Wu, S. Li, C.-J. Hsieh, J. Sharpnack, Sse-pt: Sequential recommendation via personalized transformer, in: Proceedings of the 14th ACM Conference on Recommender Systems, RecSys '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 328–337.
[42] J. Dodge, S. Gururangan, D. Card, R. Schwartz, N.A. Smith, Show your work: Improved reporting of experimental results, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, 2019, pp. 2185–2194.
[43] Y. Tian, J. Ma, Q. Gong, S. Sengupta, Z. Chen, J. Pinkerton, L. Zitnick, Elf opengo: An analysis and open reimplementation of alphazero, in: International Conference on Machine Learning, PMLR, 2019, pp. 6244–6253.
[44] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I.D. Raji, T. Gebru, Model cards for model reporting, in: Proceedings of the conference on fairness, accountability, and transparency, 2019, pp. 220–229.

Ge Gao is a Ph.D. candidate in School of Economics and Management, Tsinghua University. He earned his Bachelor in Dalian University of Technology. His research focuses on information systems and data mining. His current research is about the design of recommender systems with the consideration of social and psychological factors.

Hongyan Liu is a professor at School of Economics and Management, Tsinghua University. She received her PhD degree in management science from Tsinghua University. Her current research interests include data/text mining, machine learning, personalized recommendation, social computing and computer vision. She has published many papers in journals such as MISQ, ISR, INFORMS JOC, ACM TODS, ACM TOIS, and IEEE TKDE, and in conferences such as VLDB, ICDE, SIGKDD, ICDM, SDM, CIKM and ICIS.

Kang Zhao is an Associate Professor of Business Analytics at Tippie College of Business, The University of Iowa. His current research focuses on data science and social computing, especially the mining, predictive modeling, and simulation of social, business, and collaboration networks. His research has been featured in public media from more than 25 countries. He also served as the Chair for INFORMS Artificial Intelligence Section 2014–2016. He earned his Ph.D. from Penn State University.