Received May 30, 2020, accepted June 14, 2020, date of publication July 9, 2020, date of current version July 28, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3007675

Facing Cold-Start: A Live TV Recommender System Based on Neural Networks

XIAOSONG ZHU[1,2], JINGFENG GUO[1,2], SHUANG LI[3,4,5], AND TONG HAO[1,2]
[1]College of Information Science and Engineering, Yanshan University, Qinhuangdao 066004, China
[2]The Technology Innovation Center of Cultural Tourism Big Data of Hebei Province, Chengde 067000, China
[3]Faculty of Ecology, Environmental Management College of China, Qinhuangdao 066102, China
[4]School of Architecture, Tianjin University, Tianjin 300072, China
[5]Key Laboratory of Urban Landscape Ecology & Planning and Design of Qinhuangdao, Qinhuangdao 066102, China

Corresponding author: Jingfeng Guo (tanxunqixidi@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61472340, and in part by the Hebei Provincial Department of Science and Technology under Grant 20310301D.

ABSTRACT With the increase in the number of live TV channels, audiences must spend increasing amounts of time and energy deciding which shows to watch; this problem is called information overload, and recommender systems (RSs) are effective methods for addressing such problems. Due to the high update rates and low replay rates of TV programs, the item cold-start problem is prominent, and this problem seriously affects the effectiveness of the recommender and limits the application of recommendation algorithms for live TV. To solve this problem better, RSs must consider information in addition to the time slot strategy, which relies on experience. At present, no methods make good use of viewing behavior records. Therefore, in this paper, we proposed a viewing environment model called DeepTV that considers viewing behavior records and electronic program guides and includes a feature generation process and a model construction process. In the feature generation process, we defined seven key features by clustering viewing time, distinguishing positive and negative feedback, capturing continuous viewing preference and introducing the remaining time proportion of candidate programs. We normalize the continuous features and add powers of them. In the model construction process, we regard the live TV recommendation task as a classification problem and fuse the above features by using a neural network. Finally, experiments on industrial datasets show that the proposed model significantly outperforms baseline algorithms.

INDEX TERMS Cold start, live TV, neural network, negative feedback, TV channel, viewing environment.

I. INTRODUCTION

For a long time, live TV was the primary means of watching TV programs. With the increase in the numbers of TV channels and TV programs, the audience has to spend increasing amounts of time and energy deciding which programs to watch; this problem is referred to as information overload. As an effective method for addressing information overload [1], recommender systems (RSs) have been widely studied and applied in academia and industry. Among such systems, the live TV RS has been developed for commercial applications in some regions of North America [2], [3] and Europe [4]. In recent years, with the increase in the penetration rate of smart TVs and various smart set-top boxes, RSs have been widely applied for live TV.

Live TV differs from Video on Demand (e.g., YouTube [5], Netflix [6]) in several ways that make the research of its RSs difficult. First, every day, many programs are shown on live TV that have not been broadcast in the past; thus, there are no viewing records (scores) [7]. RSs cannot recommend new programs due to the cold-start [8] problem. Second, families constitute the main audience of live TV, and multiple family members share a terminal [9], [10]; thus, an RS needs to identify the preferences of multiple people. Third, an audience shows its preference for a program via implicit feedback (e.g., viewing duration) rather than explicit feedback (e.g., rating). The length of viewing duration implies the preference for a video. For example, watching a program for a few seconds may mean that a user does not like the current program (i.e., negative feedback), while the opposite behavior may indicate positive feedback. Therefore, an RS should utilize implicit feedback. Fourth, live TV is a real-time service, and a

The associate editor coordinating the review of this manuscript and approving it for publication was Vicente Alarcon-Aquino.

recommendation can only be given when candidate programs are being broadcast [11]. Thus, the time window for live TV recommendation is narrow, and the RS needs to respond quickly to user feedback.

In recent years, live TV RSs have become a popular research topic in the recommendation field. Live TV programs take the channel as a transmission carrier, and the recommendation models rely on both the $<$ user, program, rating $>$ and the $<$ user, channel, rating $>$. Therefore, we divided related research into methods for recommending TV programs (RP methods) and method for recommending TV channels (RC methods) according to the recommended content. The most common RP method is a collaborative filtering-based algorithm (CF) that constructs a *user-program* rating matrix [12]–[14] or a *user-program-other* tensor [10], [15] and uses the CF to predict preferences. However, CF cannot cope with the cold-start problem. Content-based (CB) algorithms [16]–[19], context-based algorithms [20], and social network-based algorithms [21], [22] can address such problems, but these algorithms require different amounts of additional information. RC methods rely on the time factor to establish a *channel-time* correlation, and they convert the program preferences of users to the channel preferences of users [7], [9], [11], [23]–[29]. Such methods can handle the cold-start problem and have good real-time performance. However, the above models have the following disadvantages: (*W1*) they identify user preferences by considering time; however, time division relies on experience, and it is not universal or interpretable; (*W2*) they do not consider the characteristics of candidate programs at the recommended time; thus, they cannot adjust the prediction according to these programs; and (*W3*) for RP and RC methods, either positive and negative feedback are not differentiates or negative feedback is ignored.

We focus on the item cold-start problem of live TV by using user logs and electronic program guides (EPG) to address the disadvantages (*W1*, *W2*, *W3*) of existing RC methods, and a v*iewing-environment*-based model that includes a *feature generation* process and a *model construction* process is proposed (Fig.1). During *feature generation*, we define seven key features by clustering viewing time, distinguishing positive and negative feedback, capturing continuous viewing preferences and introducing the proportion of time remaining in candidate programs. We normalize the continuous features and add powers of them. During *model construction*, we regard the live TV recommendation task as a classification problem and fuse the above features using a neural network. These features are not associated with the user feedback on candidate programs at the recommended time; thus, the cold-start problem can be addressed. To distinguish our method from other strategies, the factors that produce these features are referred to as the *viewing environment*.

The main contributions of this paper include the following:

(1) We divide each day into several time slots by clustering the start time of watching behavior, and periodicity is considered.

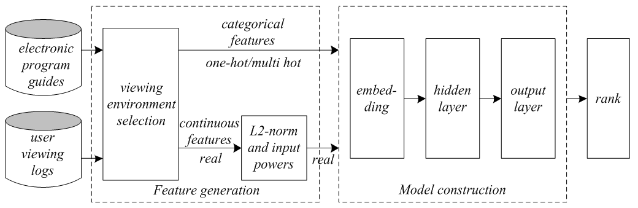


FIGURE 1. The overview of our work.

(2) We divide implicit feedback into positive feedback and negative feedback, and we use them separately.

(3) We define *viewing heat*, *viewing heat in the current time slot* and *viewing distance* to reflect the continuous viewing preference of the user.

(4) We correlate the candidate programs with the RS model by introducing the proportion of time remaining for candidate programs.

(5) The live TV recommendation task under the cold-start scenario is regarded as a classification-based sorting task, and a recommendation model is constructed using a neural network.

II. RELATED WORK

This paper considers the live TV recommendation task under an item cold-start scenario. Next, we present recent studies from the related literature and divide these studies into RP methods and RC methods. In addition, because the model in this paper uses a neural network, some applications of deep learning for addressing the cold-start problems of RSs are introduced.

A. PROGRAM RECOMMENDATION METHODS

RP methods mainly include traditional CB and CF methods. In 2015, Ras *et al.* [30] surveyed the literature on TV RSs, with a focus on RP methods. CB algorithms are commonly used to solve the cold-start problem in recommendation tasks. Bambini *et al.* [16] recommended new programs to users by calculating the similarity of media information between programs. Li *et al.* [19] adopted a CB method by calculating the similarity between users and programs to acquire a prediction, and adopted a collaborative filtering process of CF-SVD-CF to acquire the other prediction, and acquired the final result by predictions fusion. This method used the CB section to alleviate the item cold-start problem. Hölbling *et al.* [18] used a CB method to present an effective and flexible tag generation process. The recommendation results of CB are poor in terms of novelty and diversity.

The application of CF for live TV is similar to that in VOD. Hu *et al.* [14] converted user preferences for programs to 0 or 1 and used matrix decomposition to predict user preferences. Jin *et al.* [12] constructed a rating function with viewing duration and viewing duration proportion and used Pearson's correlation coefficient to implement user-based CF. Wang *et al.* [10] proposed an algorithm that determines the time slots for each account by clustering the factorized time subspace, and similar activities among these slots are combined to represent members. Cho *et al.* [13] used weighted alternating least squares to predict the preference in a user-item preference matrix to generate the final rating matrix and adjusted the rating matrix by using a view probability model. RecTime [15] employs 4D tensor factorization, which considers two additional time factor dimensions. By factorizing the 4D tensor, the system naturally identifies both the recommendation time and the items. However, CF has to address the matrix sparsity and program cold-start problems.

Other methods, e.g., Change *et al.* [21] and Zhang *et al.* [22], derived user preferences from user viewing history and social networks. Kim *et al.* [31] applied a knowledge extension method to media recommendation based on media information. Based on metadata, Philipp *et al.* [32] adopted a field-aware factorization machines model for publishing TV content onto secondary publication channels. Using an explicit hybrid strategy, Seo *et al.* [33] mixed OTT data with IPTV data to improve the recommendation accuracy. Hsu *et al.* [20] carried out recommendation task research by obtaining numerous user attributes, including interest, emotion, experience, population information, etc., and implemented program recommendations by using ANN. The above methods rely on different types of data to solve the cold-start problem, and the recommendation effect depends on the dimension [34] and quality of the data. Note that for these methods, operators must obtain relevant data, which is challenging. In addition, by employing statistics-based methods, ShowTime [35] constructs user-item preference matrix and viewing frequency matrix to predict candidate programs. ShowTime focuses on a user's preference for a program and does not rely on additional information; thus, it cannot address the cold-start problem.

In fact, a TV program is broadcast on a TV channel according to a schedule, and it is challenging to determine how suitable a program is for a viewer when the viewer changes the channel [37].

B. TV CHANNEL RECOMMENDATION METHOD

The RC method is a unique solution for the live TV recommendation task. Similar to live streaming, a channel recommender, instead of a program recommender, is more suitable for live broadcasts. It converts a user's preference for TV programs into the preference for TV channels and thus can be used to bridge historical TV programs with new ones. Most of these methods used the time factor to divide a TV channel into several logical channels or to divide a shared account into several virtual viewers to achieve preference differentiation. The scale of the preference estimation model (e.g., *user-channel-time*) is much smaller than that of the *user-item* matrix; thus, it can alleviate the problem of data sparseness. For example, Cremonesi *et al.* [9] divided a day into eight time slots to build a *time-channel* rating matrix for each user, used the cumulative viewing duration of each time slots as the user viewing preference in that slot, and realized preference prediction by applying Tucker decomposition. Turrin *et al.* [7], Wu *et al.* [25] and Kim *et al.* [29] divided a week into several time slots to capture user preferences in combination with TV program metadata. However, the time-division strategies of the above methods rely on experience and have poor universality, as they only focus on the channel and ignore the features of the current programs at the time of recommendation.

Yu *et al.* [23] proposed six channel-based recommendation strategies. By considering recommendation as a binary classification task, they integrated the above recommendation strategies. This method considered a user's preference for a channel and the current programs being broadcast, but it ignored negative feedback. Zui *et al.* [26] provided a hybrid preference-aware recommendation algorithm that adopted the cumulative duration spend viewing a channel as the channel preference. Bahn and Baek [27] analyzed user viewing behaviors by channel to recommend channels. Ning *et al.* [11] discussed the difficulty of live channel recommendation and proposed ways to intelligently recommend channels to users. Lin *et al.* [28] proposed the user preference clustering algorithm to solve the channel recommendation problem for live game streaming platforms, channel/game/language preference calculation, clustering computation, and channel score calculation. However, the above methods did not distinguish positive feedback from negative feedback. Except for [23], the studies did not consider the characteristics of candidate programs at the recommended time.

C. RECENT DEEP LEARNING RESEARCH ON THE COLD-START PROBLEM

With the application of deep learning in RS, some internet companies adopted neural networks to address the cold-start problem [38]–[40]. Alibaba uses the embedding of side information in new items to replace the embedding of the items to calculate the similarity between items [38], [39]. For the cold-start problem of housing options, Airbnb selects the three closest houses of the same type (within a radius of 10 miles) and uses the average value of the embedding of the three houses as the embedding of the new house [40]. For new music, Oramas *et al.* [41] extracted semantic information and text features from the author's introduction and extracted track embedding information from music signals. Such methods show that deep learning has a strong ability to make use of heterogeneous data.

In summary, in the program recommendation method, the CB has a single recommendation type and an unsatisfactory recommendation effect, while the CF cannot address the matrix sparsity and cold-start problems produces poor real-time recommendations. The context-based approach and the social network-based approach rely on multidimensional datasets, which increases the difficulty of implementation. For the RC method, the time division rules are not universal and have poor interpretability. Most methods do not focus on the current programs at the time of recommendation. In addition, the two methods cannot effectively distinguishing positive feedback from negative feedback.

Therefore, while employing no additional information, this paper aims to address the cold-start problem of live TV programs and improve recommendation effect by optimizing the time-division strategy, considering the candidate programs at the time of recommendation, using positive and negative feedback and capturing continuous viewing preferences. Due to the advantages of deep learning for information utilization, in this paper, we adopt deep learning methods to fuse heterogeneous data from various features.

III. PRELIMINARIES

A. TASK DEFINITION

We first define our research task. The user set is U, the TV channel set is C, and the historical set of TV programs is $R^{history}$. At time t, the set of TV programs that are on air is R^{now}, and the cold-start problem of the TV program is represented as $R^{history} \cap R^{now} = \emptyset$, $|R^{now}| = |C|$. For each historical TV program $p^h \in R^{history}$, argChannel(p^h) $\in C$. For each live TV program $p^n \in R^{now}$, argChannel(p^n) $\in C$. The task is to generate a recommendation list rank$_u$(C,K) (*RC* method) or $rank_u$ (R^{now}, K) (RP method) for user u at time t with length K.

If the above task is regarded as a regression problem, then the implicit feedback (viewing duration) should be transformed into explicit feedback (rating). Existing conversion methods employ the accumulation of viewing duration [35], the 0-1 method [14], [42], the proportion of TV viewing duration [1] and the optimized proportion of TV viewing duration (state-of-the-art) [13]. However, this conversion process inevitably loses some information, and it can easily introduce error. For instance, the optimized proportion of TV viewing duration adopts the ratio of user's viewing duration to program's duration the user can view. However, the denominator contains advertisements and cast which users may not watch, those contents participate in the conversion of implicit feedback to explicit feedback, and error is introduced. The ratio of advertisements (and cast) to the whole TV program are various in different TV programs, it means that the conversion errors of different TV programs are not same, so it is not ideal to regard the recommender system as a regression task with conversion results of implicit feedbacks as labels. Inspired by [5], we regard live TV recommendation as a multiclassification problem, with each candidate TV program as a classification.

In *viewing environment E*, the probability that the content w_t watched by user u at time t of TV program p is:

$$P(w_t = p|u, E) = \frac{e^{v_p}}{\sum_{i \in R^{now}} e^{v_i}}, \quad p \in R^{now} \tag{1}$$

where v_i is the output of TV program i.

The time of recommendation is critical to user experience. As shown in Fig.2, Oh *et al.* [35] evaluated the condition of recommendation every 10 minutes. The recommendation will be triggered when the rating of a TV program that is on air is higher than that of the current TV program watched by the user. Obviously, the recommendation will interrupt the user's viewing process and thus affect user experience. A more appropriate time for recommendation is the moment at which the user switches the channel. This operation indicates that the current user is looking for preferred content and that a recommendation at this time meets the user's needs; this moment is the time at which the user starts viewing programs.

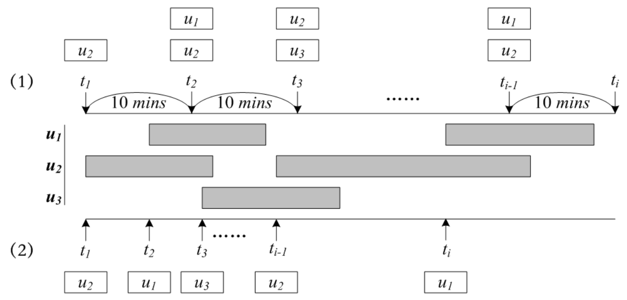


FIGURE 2. Recommendation moment. (1) Fixed duration, e.g., 10 minutes; (2) the moment the TV channel is changed.

B. OUR OBSERVATIONS

For live TV, common datasets include viewing records (Table 1) and EPG (Table 2). On this basis, this paper proposes a solution to the program cold-start problem based on the viewing environment. In Table 1, the audience's preference for a TV program is expressed as the viewing duration (*end_time* − *start_time*). Watching a TV program for a short duration indicates that the user does not like the program; this type of behavior is considered *negative feedback*; the opposite type of viewing behavior is considered *positive feedback*.

TABLE 1. View behavior logs.

Name	Description
u_id	user id
start_time	the start time of a watching behavior
end_time	the end time of a watching behavior
channel_id	TV channel id
event_id	TV program id

TABLE 2. EPG.

Name	Description
event_id	TV program id
event_name	the name of TV program
bs_time	the start time of a TV program
be_time	the end time of a TV program
channel_id	TV channel id
channel_name	channel name

How can we distinguish *positive feedback* from *negative feedback*? Cho *et al.* [13] set a threshold for the *viewing duration ratio* ($\alpha = 0.1$). The *viewing duration ratio* is defined as $r = w/T_l$, where w is the *viewing duration*, i.e., $w =$ *end_time-start_time*, and T_l is the *remaining time*, i.e., $T_l =$ *be_time-start_time*. However, -when $T_l = 50$ minutes, the conclusion that a *viewing duration* of 5 minutes

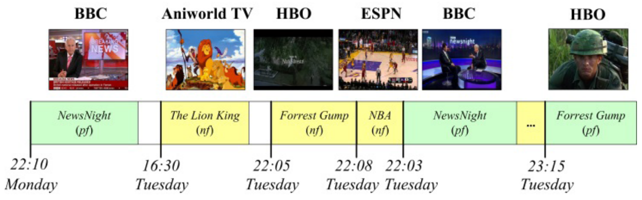


FIGURE 3. **User's viewing sequence.**

(50 minutes $\times$ 0.1) is *negative feedback* is unconvincing. Wang *et al.* [10] set a *threshold of viewing duration* (β = 300 seconds). Obviously, this method is not suitable for short TV programs. Statistics of our dataset show that a *remaining time* of more than 50 minutes accounts for 17.6% of all viewing behavior, while a *remaining time* less than 300 seconds accounts for 14.0% of all viewing behavior; thus, the errors caused by the two methods cannot be ignored. To define *negative feedback* more precisely, we combined both rules, and the viewing behaviors meeting the conditions $\alpha < w/T_l$, and $w < \beta$ are considered as *negative feedback*.

Here we discuss the recommendation task of live TV, which is a common viewing scenario, and only the programs being broadcasted are regarded as candidates. The PVR and TV replaying functions are not considered. Suppose L_u is a viewing sequence of user u with positive feedback and negative feedback (Fig.3). In L_u, user u watched *News Night* (positive feedback) at 22:10 on Monday. On Tuesday, he started to watch *The Lion King, Forrest Gump* and the *NBA*, respectively from different channel, at 16:30 (negative feedback). At 22:03, he switched to a news channel to watch the *National News* (positive feedback). We noticed that:

OB1: 22:03 and 22:10 are likely to belong to the same viewing time slot for user u.

OB2: *The Lion King*, *Forrest Gump* and the *NBA* should not be recommended to user u at 22:03 on Tuesday because user u has provided negative feedback. Since *The Lion King* finished before 22:03, the program aired subsequently can be used as a candidate for user u. To this end, we adopted the methods proposed in [13] and [10] respectively to distinguish the positive feedback and negative feedback to calculate the probabilities, which are an on-aired program p viewed as a negative program by user u will be re-watched by user u as a positive program (e.g., the probability of watching *Forrest Gump* at 23:15 by user u), the results are 19.51% and 34.99% respectively, while the probability of program p will be re-watched as the first positive program after continuous negative feedbacks (e.g., the probability of watching *Forrest Gump* at 22:03 user u) are only 2.04% and 2.49%, the statistics confirmed our inference.

OB3: The viewing behavior (*National News*) for which *positive feedback* was provided on Monday has a positive influence on the viewing behavior on Tuesday.

OB4: When user u starts to watch *News Night*, a significantly amount of its content of *News Night* remains to be broadcast because user u is reluctant to watch a TV program that is almost over.

To make use of the above information, traditional machine learning methods need to concatenate different models. Such methods are difficult to implement, and acquiring a good effect is difficult.

In our work, the state of TV programs that are on air at time t and the historical viewing behaviors that may affect the viewing behaviors of user u at time t are called the *viewing environment* of user u. For user u, suppose that the probability of viewing a historical TV program p^h under environment E is P, and the probability of viewing a TV program p^n that is on air under environment E' is P'. If $P = F(u, p^h, E)$, then $P' = F(u, p^h, E')$. Next, we will present the feature generation process based on the *viewing environment*.

IV. FEATURE GENERATION

In this section, we try to extract features from our observations and express them in a continuous or discrete way. Continuous features are represented by real numbers [5], while discrete features are represented by one-hots or multi-hots. This part includes two components: feature selection and feature processing. We first define seven key features.

A. USER ID

We extract *u_id* without considering *event_id*. As mentioned in part II, historical TV programs will not be replayed in the future; thus, it is meaningless to extract *event_id* from the historical record.

B. THE START TIME OF POSITIVE FEEDBACK

We hope to make recommendations at the moment t when the user selects a channel, which corresponds to the *start_time* (Table 1). Therefore, we regard *start_time* as a key feature. In general, live TV usually serves the whole family, and the *start_time* often varies by family member. For example, housewives and older people prefer to watch TV in the morning, children watch in the afternoon after school, and other adults watch in the evening or at midnight (***OB1***). Therefore, we divide each day into ts time slots so that the *start_time* of each set of viewing behaviors belongs to one time slot only.

$$D := \bigcup_{s=1}^{ts} d_s \quad s.t. \bigcap_{s=1}^{ts} d_s = \emptyset \tag{2}$$

Equation (2) shows that viewing cycle D (24 hours per day) is composed of consecutive time slots that do not intersect; d_s represents the s-th slot. Each *start_time* belongs to a corresponding slot d_s and is transformed into discrete features represented by a one-hot.

Definition 1: The time slot of positive feedback

Positive feedback of user u occur at time t is expressed as $PF = \{pf_1, pf_2, \ldots, pf_s, \ldots, pf_{ts}\}$.

$$pf_s = \begin{cases} 1, & t \in d_s \\ 0, & else \end{cases} \tag{3}$$

During the test phase, we used such feedback as the input of the model. For example, for the recommendation task of user u at time t', we used time t' as the *start_time* to generate *PF*.

To implement time division, the *start_time* values of all sets of user viewing behaviors are taken as the inputs to K-means++, and *ts* time slots are clustered by calculating the Euclidean distance between each pair of *start_time* values. The result is shown in Fig.4 as an example.

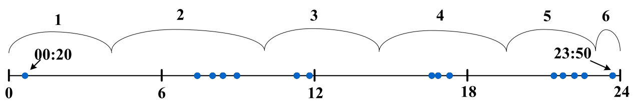


FIGURE 4. Time feature clustering.

Note that the time characteristics are cyclical. For example, *start_time* values of 00:20 and 23:50 are likely to be in the same slot for user u (e.g., an adult). However, if we do not consider the cyclic characteristics of time, they may belong to period 1 and period 6 after clustering (Fig.4).

The one-dimensional feature of time is usually mapped to two-dimensional features by Equation (4) [43]. As an example, the mapping effect is shown in Fig.5-a).

$$\begin{cases} x_{sin} = sin(\frac{2\pi t}{24}) \\ y_{cos} = cos(\frac{2\pi t}{24}) \end{cases} \tag{4}$$

Here, t is the *start_time*. However, the Euclidean distance between any two points cannot reflect the real difference between two *start_time* values. As shown in Fig.5-a), the distance from 0 o'clock to 6 o'clock is obviously not 6 times the distance from 0 o'clock to 1 o'clock. Therefore, we map time to a circle with a radius $r = 1/2\pi$; for example, the result for time $a = 13{:}30(13.5)$ is $a' = 2\pi ra/24 = a/24 = 13.5/24$. The minimum distance on the arc of the two points (time) is taken as the distance between them for clustering, i.e., the distance of time a and time b is $d_{ab} = min(|a'-b'|, 2\pi r\text{-}|a'-b'|)$. Then, the clustering effect becomes as shown in Fig.5-b), and slot 1 and slot 6 are combined into one slot.

C. FEATURE EXTRACTION OF NEGATIVE FEEDBACK

Inspired by ***OB2***, *negative feedback* shows that user u does not like the TV program or that it is not worth watching as it nears the end; TV programs that are airing and have been watched but are disliked by user u should not appear in the recommendation list of user u. For user u, the TV channel, which is still broadcasting the TV program that received negative feedback at time t (recommended time) is regarded as the input of the neural network. The input is represented by a multi-hot.

Suppose that *positive feedback* is provided at time *t−1* and time *t,* and the *negative feedback* of all channels at time t is expressed as $NW = \{nw_1, nw_2,\ldots, nw\ c,\ldots, nw_n\}$.

$$nw_c = \begin{cases} 1, & if\ p_c \in ch_c \\ 0, & else \end{cases} \tag{5}$$

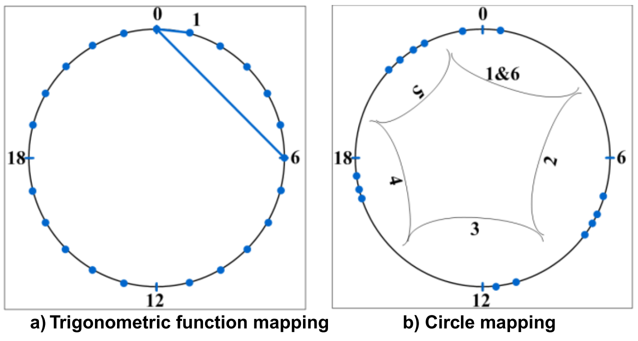


FIGURE 5. Mapping method of time-periodic characteristics.

Here, $p_c \in I_u \cap I_t$, I_u represents the TV programs for which user u provided negative feedback from time t to time $t-1$, and I_t represents the TV programs airing at time t.

D. CONTINUOUS VIEWING PREFERENCE

The *viewing behavior* of user u before the recommended time t reflects user u's preference at time t to some extent. For example, user u likes to watch programs (e.g., news, TV shows, variety shows) on the same channel and at the same time (***OB3***). Therefore, we map these features to the corresponding TV channel $CH = \{ch_1, ch_2,\ldots, ch_c,\ldots, ch_n\}$.

Definition 2: Viewing heat

Within h hours before time t, user u shows the *positive feedback* of each channel, i.e., $BW = \{bw_1, bw_2,\ldots, bw_c,\ldots, bw_n\}$.We set $h = 24$ in the following experiments.

$$bw_c = \begin{cases} 0, & if\ I_{cth} = \emptyset \\ |I_{cth}|, & else \end{cases} \tag{6}$$

Here, I_{cth} represents the *positive feedback* of user u on channel C within h hours before time t.

Definition 3: Viewing heat of the current period

Within h hours before time t, user u provides *positive feedback* for each channel in d_s, $t \in d_s$, i.e., $FW = \{fw_1, fw_2,\ldots, fw_c,\ldots, fw_n\}$. We set $h = 24$ in the following experiments.

$$fw_c = \begin{cases} 0, & if\ I_{cd_sh} = \emptyset \\ |I_{cd_sh}|, & else \end{cases} \tag{7}$$

Here, I_{cd_sh} represents the *positive feedback* of user u for channel C within h hours before time $t(t \in d_s)$, and the *positive feedback* is provided in d_s.

Definition 4: Viewing distance

For user u, the number of days between *start_time* t' of the last *positive feedback* provided for each channel and time t, t' *and* $t \in d_s$, i.e., $DW = \{dw_1, dw_2,\ldots, dw_c,\ldots, dw_n\}$.

$$dw_c = \begin{cases} 0, & if\ I_{cd_s} = \emptyset \\ t - max(T_{cd_s}), & else \end{cases} \tag{8}$$

Here, T_{cd_s} indicates the *start_time* set of all *positive feedback* of TV channel C, and each *start_time* belongs to d_s.

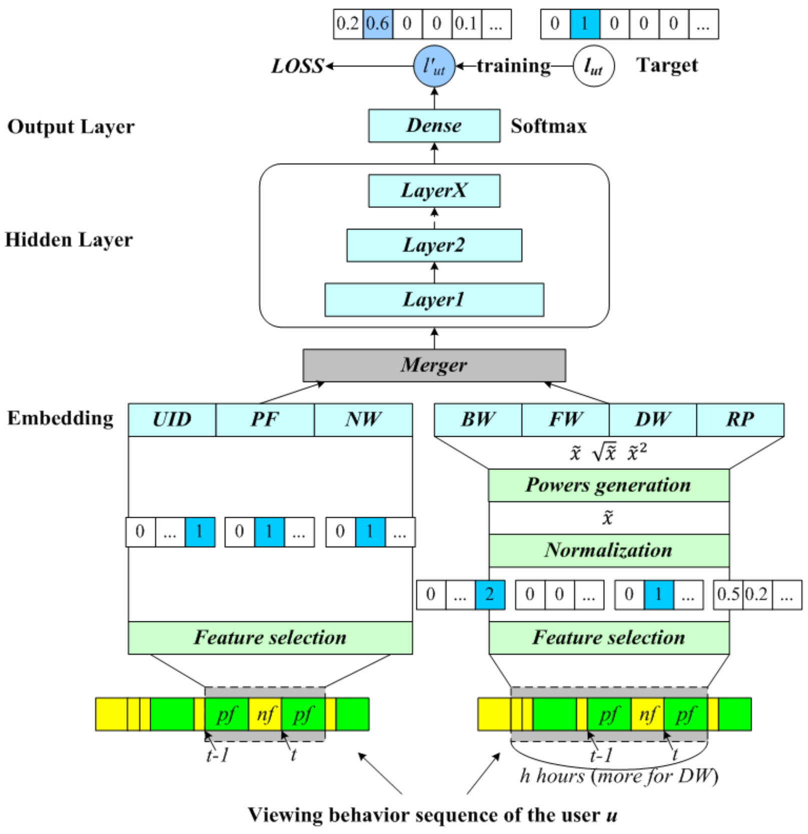


FIGURE 6. The structure of DeepTV (*pf* means positive feedback while *nf* means negative feedback).

E. FEATURES OF THE AIRING TV PROGRAM

To establish a direct relationship between the new TV programs and the model, inspired by OB4, we introduced the proportion of time remaining [35].

Definition 5: The proportion of time remaining

At time t, the *proportion of time remaining* of TV program p in channel C is:

$$rp_c = \frac{be\_time_p - t}{be\_time_p - bs\_time_p} \tag{9}$$

where be_time_p is the be_time of p, and bs_time_p is the bs_time of p. At time t, the *proportion of time remaining* of all TV channels is expressed as $RP = \{rp_1, rp_2, \ldots, rp_c, \ldots, rp_n\}$.

F. FEATURE PROCESSING

Before being fed to the neural network, the continuous features are first treated as $\tilde{x}$ by the L2 norm, and $\sqrt{\tilde{x}}$ and $\tilde{x}^2$ are added to provide the neural network with more expression ability. The continuous features include *viewing heat*, *viewing heat of the current period*, *viewing distance*, and the *proportion of remaining time*. This process improves the accuracy of offline training [5]. Then, all features are embedded through a fully connected layer. The dimension e_q of the embedding layer is in $[1, f_q-1]$. f_q is the number of classifications of the feature, and e_q is selected based on experience, as there is no relevant theory related to its determination [44]. The embedding maps the nonzero term of a one-hot or multi-hot to the connection weight corresponding to the full connection, and then it realizes the transformation from input to network expression.

V. MODEL CONSTRUCTION

Our model, **DeepTV**, consists of a feature generation process and a model construction process (Fig.6). Feature generation includes feature selection, normalization and powers generation. Model construction includes input, hidden and output layers. The input layer is a fully connected layer for embedding the input features. In the hidden layer, a suitable neural network is used for network fitting. The output layer is a fully connected layer that takes softmax as the activation function. In the classification task, cross entropy is usually chosen as the loss function.

$$Loss = -\sum_{c=1}^{|C|} y_c^i \log(P_c^i) \tag{10}$$

Here, y_c^i represents whether the c-th sample belongs to the $i-th$ category, with a value of 0 or 1. P_c^i represents the probability that the model predicts that the $c-th$ sample belongs to the $i-th$ category. However, *cross entropy* is not suitable for live TV. For user u, the candidates are $l = \{c_1,$

$c_2, c_3, c_4\}$,the correct answer is c_2, and we use *cross entropy* as the loss function. Suppose that there are three outputs,i.e., $l_1 = \{c_1{:}0.4, c_2{:}0.3, c_3{:}0.1, c_4{:}0.2\}$, $l_2 = \{c_1{:}0.32, c_2{:}0.30, c_3{:}0.31, c_4{:}0.07\}$, $l_3 = \{c_1{:}0.0, c_2{:}1.0, c_3{:}0.0, c_4{:}0.0\}$; thus, we obtain the corresponding losses: $loss_1 = 0.52$, $loss_2 = 0.52$, and $loss_3 = 0$. Although l_1 and l_2 are wrong, l_1 (with c_2 in second place) is better than l_2 (with c_2 in third place). However, the loss function does not reflect their difference ($loss_1 = loss_2$). Therefore, *cross entropy* cannot reflect the training state of the neural network. In l_3, although the network converges, it sacrifices efficiency and increases the risk of overfitting. Therefore, we propose a position-dependent loss function:

$$Loss = \frac{1}{|L|}\sum_{l \in L} \frac{Index_l(p)}{\log_2(Index_l(p)+1)} - 1 \tag{11}$$

Where $Index_l(p)$ represents the position of the correct answer p in the candidate list l, and L is the set of recommended tasks.

VI. EXPERIMENTAL RESULTS AND ANALYSIS

Our dataset includes the user viewing record and EPG from May 17, 2017 to June 20, 2017. The dataset contains 30,187,636 records of 35,143 set-top boxes. In order to test the effect of our proposal on item cold-start problem, we randomly selected 500 accounts, and each account had at least 200 viewing records. Each user's viewing record is sorted according to the *start_time* and divided into a training set and a testing set with a ratio of 80:20. Each *event_id* is different, and all algorithms are implemented by assuming the cold start of the TV program, i.e., $R^{history} \cap R^{now} = \emptyset$. We adopt recall, normalized discounted cumulative gain (nDCG), and mean reciprocal rank (MRR) to evaluate the prediction performance.

To show the performance of our algorithm, we select some recent TV recommendation algorithms with different technologies as comparison methods and adopt the best parameters given in these literatures. All comparison methods and our algorithm adopt the same viewing records (Table 1) and EPG (Table 2). Some methods are not chosen because of the distinct datasets they need, e.g., the methods based on the content and the methods based on social network. In addition, we add *the most popular* method to reflect the viewing characteristics of live TV.

TD [9]: Time-based model. First, it divided each day into eight viewing timeslots and constructed a time-channel rating matrix for each user. Secondly, the cumulative viewing duration of each timeslot was taken as the preference of user in that period. Finally, the results were predicted by Tucker Decomposition.

ShowTime [34]: Statistics-based model. First, users' preferences were obtained by calculating the viewing duration of programs. Secondly, the stay time ratio and the remaining time ratio were defined to construct a matrix of viewing probability with 10×100 for each user. Finally, ShowTime performed the product of preference and viewing probability as predicted score of candidate.

MFHM [12]: User-based collaborative filtering model. It proposed a scoring model of implicit feedback based on the viewing duration ratio and viewing duration. User-based collaborative filtering algorithm was used to implement the recommendation with correlation coefficient. To alleviate the program cold-start problem, rating of finished content of a current program was used to replace the rating of the whole program at recommendation time.

WIM [13]: Multi-strategy aggregation model. WIM is the latest RP method, which includes preference estimation process, rating prediction process and recommendation process. First, WIM constructed a rating matrix and weighting matrix. Secondly, WIM adopted the weighted least square method to the rating matrix. Finally, the rating matrix was used as the output after adjusted by a viewing probability model.

PSG [23]: Channel-based multi-strategy aggregation model. Based on six channel-based recommendation strategies, e.g., global popularity, personal popularity, PSG considered the recommendation as a typical binary classification problem.

POP: Channel-based model. At recommendation time t, the number of people who are watching channel c is considered as the rating of channel c.

First, we adjust the parameters α, β, and ts to acquire the best input for the neural network. Then, we adjust the parameters of the neural network by impact from large to small (i.e., the hidden-layer network, the number of layers, the output dimension, dropout, and initialization), and we shuffle the input sequence of data to further improve the effect. Finally, we compare our algorithm with other algorithms. This model is implemented by TensorFlow 2.0 and Keras 2.3.1. All the algorithms related to the neural network are tested 30 times, and the average is taken as the final result.

A. INPUT CONSTRUCTION FOR NEURAL NETWORK

To facilitate the calculation, we build a simple neural network (the hidden layer is a fully connected layer, the output dimension is 512, dropout = 0.5 is suggested when there is no support of experimental results [24], and the other parameters are set to default values), and change α, β, and ts.

1) INFLUENCE OF NEGATIVE FEEDBACK

For $ts = 12$ and $\beta = 0$, we change α. As shown in Fig.7-a), nDCG@1 obtains the maximum value at $\alpha = 0.3$, but for nDCG@5, the value at $\alpha = 0.1$ is higher than that at $\alpha = 0.3$. The MRR at $\alpha = 0.1$ is slightly higher than that at $\alpha = 0.3$. We choose $\alpha = 0.1$ for the following experiments, and the result is consistent with that of [13].

The increase in α leads to a decrease in training samples that affects the generalizability of the model. With a decrease in positive feedback, negative feedback increases gradually, which leads to negative feedback samples mixed with positive feedback; thus, the quality of negative feedback decreases. Incorrect samples affect the model.

In industrial applications, the area on the TV in which the recommendation is shown is small, and the results of each

TABLE 3. Comparison of different hidden layers (dp: dropout).

Network	Recall@1	Recall@5	nDCG@1	nDCG@5	MRR	The structure of hidden layer
DeepTV_DNN(relu)	**0.4331**	**0.7145**	**0.4332**	**0.5821**	**0.5629**	512, dp
DeepTV_LSTM(sigmoid)	0.4172	0.7025	0.4173	0.5676	0.5487	256, dp
DeepTV_AE(tanh)	0.3821	0.6382	0.3823	0.5157	0.5050	1024, dp, 10, dp, 10, 1024

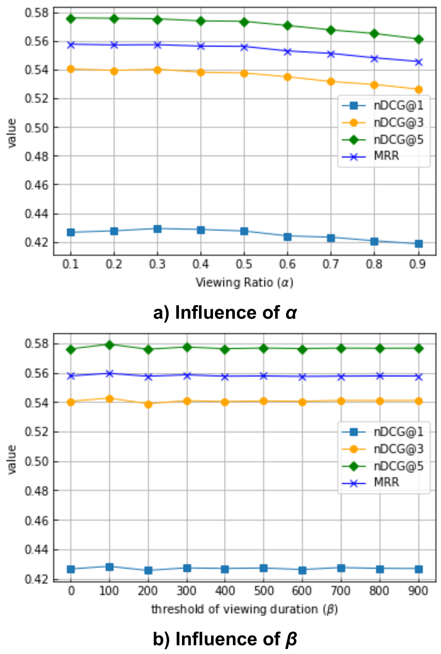


FIGURE 7. Influence of α and β.

metric at $N = 5$ is very important [45]. With a fixed $\alpha = 0.1$, we change β. As shown in Fig.7-b), the changes in nDCG and MRR are similar. The results of nDCG and MRR are optimal when $\beta = 100$.

2) INFLUENCE OF TIME DIVISION

For $\alpha = 0.1$ and $\beta = 100$, we change ts. Based on experience and [9], [10], we set the number of periods to [4,8,12,16,20,24]. As shown in Fig.8, nDCG and MRR change in similar ways, first increasing and then decreasing. When $ts = 8$, the results of each metric are optimal.

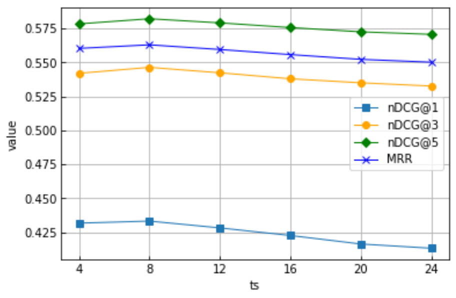


FIGURE 8. Influence of *ts*.

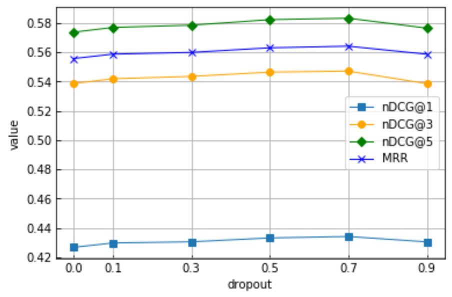


FIGURE 9. Influence of dropout.

B. NEURAL NETWORK OPTIMIZATION

1) HIDDEN LAYER SELECTION

In recent years, a variety of neural networks have been used in RSs [46]. For example, (1) before 2016, YouTube applied DNN to industrial RSs [5] for handling heterogeneous data. (2) By adding hidden-layer units to save a long-term state, LSTM can effectively model long-term dependency and is good at processing recommendation tasks for time series data [47]. (3) He *et al.* [48] extracted features from data using AE to realize collaborative filtering. We take the above three neural networks as the hidden layer to construct DeepTV, as shown in Fig.6. We set dropout = 0.5 and adopt the default method of Keras for other parameters. By changing the position of the dropout layer and the activation function, we achieve the best performance of the three neural networks after a large number of experiments (Table 3). The DNN achieves the best performance in terms of all metrics. It is a common method for training different LSTMs for each user with their data. Since there are many more parameters for LSTM than for DNN, we train a shared network with the data of all users to speed up training; to some extent, this method limits the performance of LSTM. Therefore, we use the DNN as the hidden layer in the following experiments.

2) DROPOUT

Dropout prevents overfitting the neural network by discarding the input of the neural network. Based on the previous section, we apply experiments with DeepTV_DNN (Table 3). According to Fig.9, the introduction of dropout significantly improves the effect of DeepTV. The best result is obtained for dropout = 0.7, then we re-performed experiments, and the best hidden layer did not change compared with that when dropout = 0.5.

TABLE 4. Comparison of different initialization.

Initialization	Recall@1	Recall@5	nDCG@1	nDCG@5	MRR
glorot_uniform	0.4342	0.5830	0.4341	0.7157	0.5640
he_normal	**0.4351**	**0.5838**	**0.4350**	**0.7162**	**0.5647**
he_uniform	0.4343	0.5837	0.4343	0.7157	0.5643

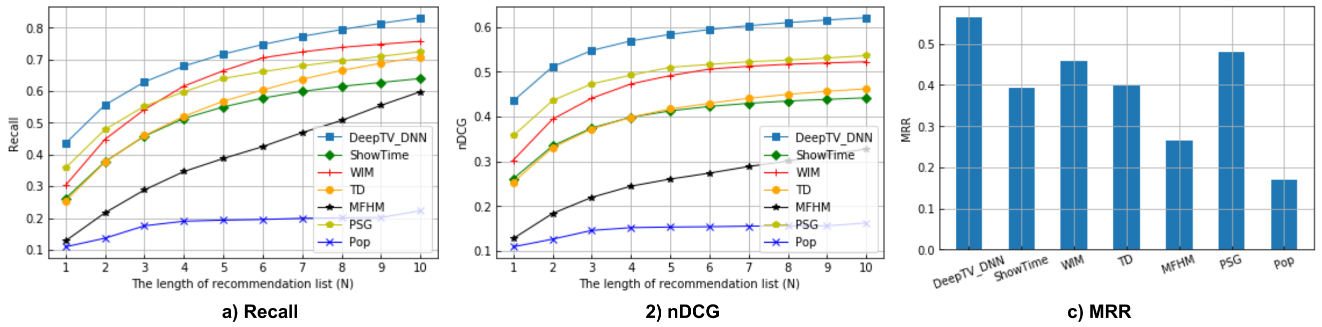


FIGURE 10. Method comparison.

3) INFLUENCE OF INITIALIZATION

When *glorot_uniform initialization* [49] is used with *relu*, the vanishing gradient problem becomes more serious with the increase in the neural network layer. Therefore, we adopt *He initialization* [36] instead of *glorot_uniform initialization*. As shown in Table 4, although DeepTV is not deep, the results are still improved after *he_normal* (*he normal distribution initialization*) was adopted.

C. DATA SHUFFLE

To avoid the influence of the sequence of input data during training, data are usually shuffled. Shuffling can prevent the model from shaking during training process and improve robustness to prevent overfitting and promote the learning of correct features. In the above experiments, the training data are put into the neural network according to *u_id*. We shuffle the input sequence of data and obtain MRR = 0.5658, which is slightly better than the result in Table 4 (0.5647). Because the data volumes of the users are similar and the user's scale is large, the possibility of local data features is low. Therefore, the effect of shuffling is not obvious.

D. COMPARISON AND ANALYSIS

In this section, we compare the proposed algorithm with existing algorithms. As shown in Fig.10, DeepTV_DNN (Table 3), the best performing method we developed, performs significantly better than the comparison algorithms. Of the comparison algorithms, WIM and PSG perform well, and the scores of each evaluation criterion are similar. With the increase in the length of the recommendation list, except for POP, the effects of other algorithms are significantly improved; MFHM is the fastest. Specifically, in terms of recall, PSG was better than WIM at first, but WIM improved faster than PSG as the length of the recommendation list increased. In terms of nDCG, PSG remained ahead of WIM in MRR. Although MFHM improved significantly with the increase in the length of the recommendation list, its performance at low N values was significantly weaker than that of other comparison algorithms, and its performance at Recall@1 was only slightly better than that of POP. POP's recommendation ability was improved at N∈[1,2,3,4,5], reaching 19.32% at Recall@5 before improvement slowed. This result indicates that several popular TV programs attract a certain number of people, but the user's demand for personalized TV programs is obvious, and TV programs show a long-tail effect. To exactly evaluate our method, we compare our method with baselines by using (recall(our method) – recall(baseline))/recall(baseline) at Recall@N∈[1,5,10], DeepTV_DNN is higher than PSG by 21.68%, 11.98% and 14.74%, higher than WIM by 43.56%, 8.93%, 9.81%, higher than TD by 71.67%, 26.09%, and 17.51%, higher than ShowTime by 66.35%, 30.29%, and 29.96%, and higher than MFHM by 240.37%, 84.92%, and 38.98%. Notably, WIM and ShowTime are unable to address the cold start of TV programs. Therefore, during preference estimation, we adopt the user preference for TV channels to replace the user preference for TV programs.

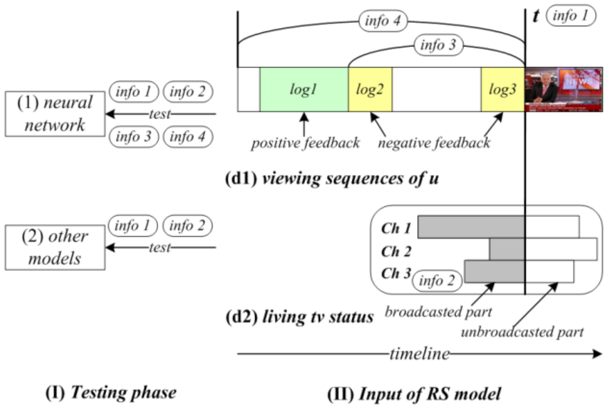
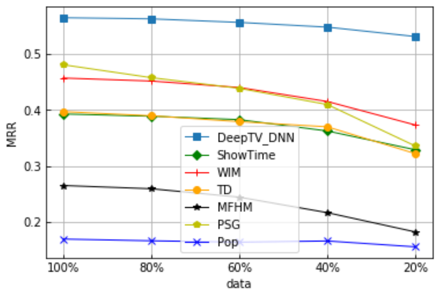
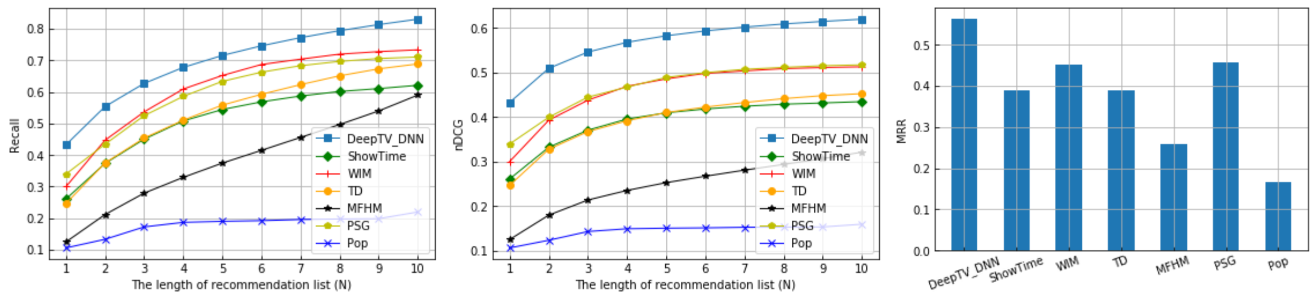


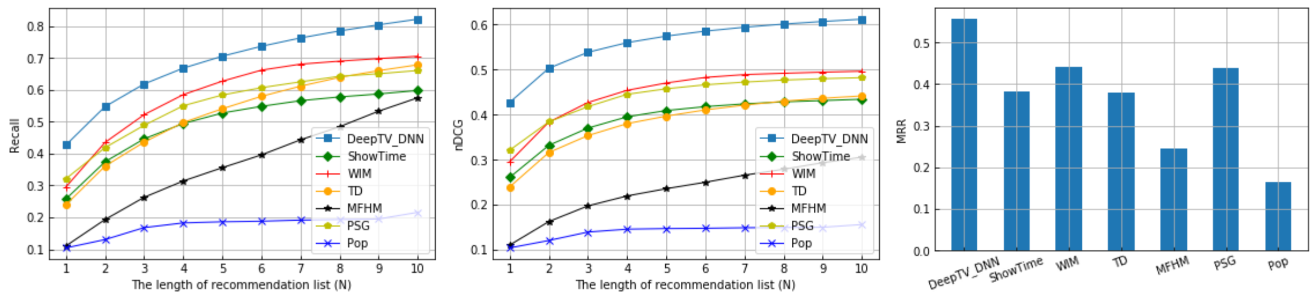
FIGURE 11. The utilization of the test set in different models.



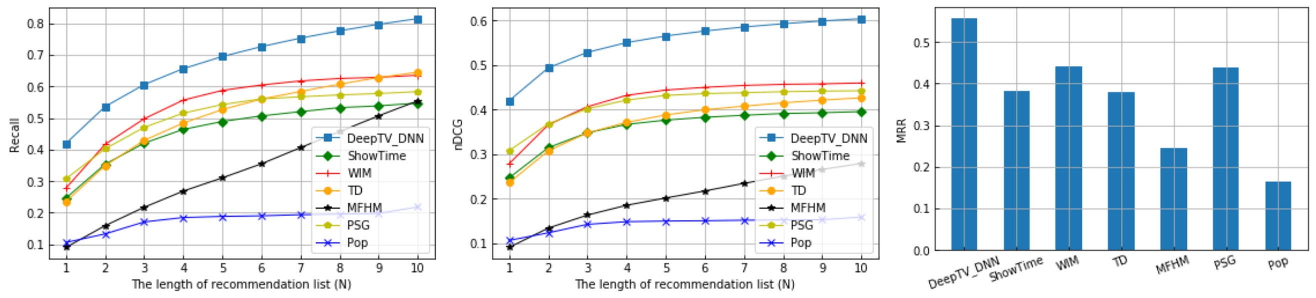
a) MRR of all algorithms for different sparse data



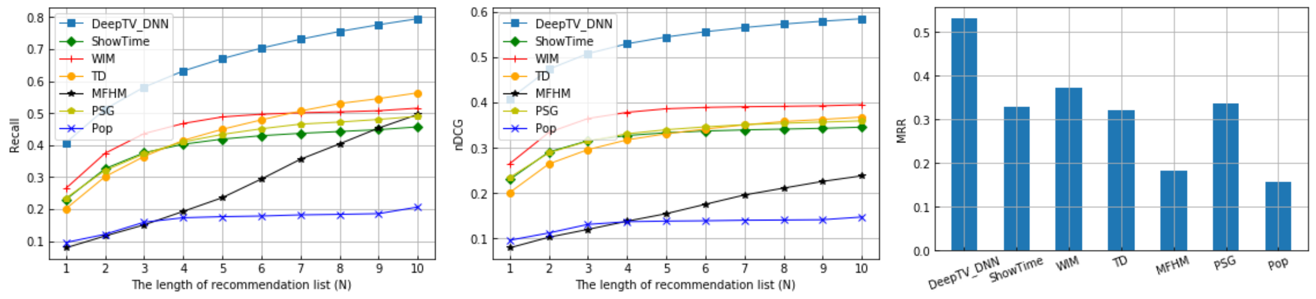
b) Recall, nDCG and MRR with 80% data



c) Recall, nDCG and MRR with 60% data



d) Recall, nDCG and MRR with 40% data



e) Recall, nDCG and MRR with 20% data

FIGURE 12. Comparison of all algorithms with sparse data.

As shown in Fig.11, DeepTV is so effective because it selects features more widely than other algorithms; it can make full use of the information in the dataset, including user-time connection, positive and negative feedback, and continuous viewing preference. However, WIM extracts features including user-channel, the TV programs on air, and the viewing state of the user at *start_time*. Although PSG adopts multi-strategy fusion, these strategies are not accurate in terms of feature extraction and ignore important information, such as positive and negative feedback. TD extracts only the user-channel-time feature. This difference is reflected in both the training stage and the testing stage. As shown in Fig.11, in (I), the input of the neural network is larger than that of other models in terms of *info3* and *info4*. Specifically, in (II), at recommendation time t, the neural network model uses the information before t.

E. THE INFLUENCE OF SPARSITY

We randomly select data from training set (test set for POP) of each user with proportions of 80%, 60%, 40%, and 20% respectively without replacement of samples. As shown in Fig.12, as the dataset becomes sparser, the quality of each algorithm's candidate list gradually and significantly declines (Fig.12-a). Our method DeepTV_DNN still performs best. Specifically, MFHM and PSG are most seriously affected by data sparseness. For a sparse dataset, the user-item rating matrix of MFHM becomes sparser and cannot be used to accurately calculate the similarity between users. PSG behaves similarly. In addition, sparse data make multi-strategy fusion of PSG difficult. TD based on tensor decomposition maps preferences to time-channel correlations in small-scale tensors (far smaller than the user-item rating matrix in MFHM). Although the reduction in the training set reduces the accuracy of preference estimation to some extent, it does not make the tensor significantly sparser. ShowTime is a statistical method, but each viewing behavior of users can provide multiple fillers for the preference matrix; thus, the preference matrix does not easily become sparser due to a decrease in viewing behavior. WIM adopts user-channel preference estimation, which has similar advantages to TD; its recommendation section method adopts the same strategy as ShowTime; thus, it can mitigate the impact of data sparseness to some extent. POP and DeepTV_DNN were slightly affected by data sparseness. Because POP adopts the method of popularity calculation, sparse data do not lose statistical significance. DeepTV_DNN benefits from richer input and stronger feature extraction ability than other algorithms.

VII. CONCLUSION

This paper focuses on the cold-start problem of live TV programs. To address the weaknesses of existing methods, we propose a recommendation model based on the viewing environment by taking the viewing record and simplified electronic program guides as the dataset. Compared with existing methods, we improved our work by (1) using clustering to divide the time slots, (2) dividing implicit feedback into positive feedback and negative feedback, (3) capturing the continuous viewing preference of users, and (4) introducing the broadcast status of candidate programs. We regard the live TV recommendation task as a classification problem and use a neural network to integrate the above features. Experiments on real datasets show that the proposed model is obviously superior to traditional live TV recommendation algorithms.

REFERENCES

[1] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005.

[2] S. Sefati, J. Neumann, and H. Sayyadi, "TV and Movie Recommendations: The Comcast Case," in *Collaborative Recommendations: Algorithms, Practical Challenges and Applications*. Singapore: World Scientific, 2018, pp. 465–479. [Online]. Available: https://www.worldscientific.com/worldscibooks/10.1142/11131

[3] K. Ali and W. van Stam, "TiVo: Making show recommendations using a distributed collaborative filtering architecture," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 394–401.

[4] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender Systems Handbook*, Berlin, Germany: Springer, 2011, pp. 1–35. [Online]. Available: http://www.springerlink.com

[5] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for YouTube recommendations," in *Proc. 10th ACM Conf. Recommender Syst.*, Boston, MA, USA, Sep. 2016, pp. 191–198.

[6] B. Hallinan and T. Striphas, "Recommended for you: The Netflix prize and the production of algorithmic culture," *New Media Soc.*, vol. 18, no. 1, pp. 117–137, Jun. 2014.

[7] R. Turrin, A. Condorelli, P. Cremonesi, and R. Pagano, "Time-based TV programs prediction," in *Proc. 1st Workshop Rec. Syst. TV Online Video ACM (RecSys)*, Silicon Valley, CA, USA, 2014, p. 7.

[8] P. Brusilovsky, A. Kobsa, and W. Nejdl, *The Adaptive Web—Methods and Strategies of Web Personalization* (Lecture Notes in Computer Science), no. 5. Jan. 2007, pp. 377–408.

[9] P. Cremonesi, P. Modica, R. Pagano, E. Rabosio, and L. Tanca, "Personalized and context-aware TV program recommendations based on implicit feedback," in *Proc. Int. Conf. Electron. Commerce Web Tech.*, Valencia, Spain, 2015, pp. 57–68.

[10] Z. Wang and L. He, "User identification for enhancing IP-TV recommendation," *Knowl.-Based Syst.*, vol. 98, pp. 68–75, Feb. 2016.

[11] L. Ning, Z. Zhao, R. Zhou, Y. Zhang, and S. Feng, "Realtime channel recommendation: Switch smartly while watching TV," in *Proc. Int. Workshop Frontiers Algorithmics*, Qingdao, China, 2016, pp. 183–193.

[12] Y. Jin, G. Junhua, Z. Suqi, and Z. Jian, "Spark-based distributed multi-features hybrid IPTV viewing implicit feedback scoring model," *Procedia Comput. Sci.*, vol. 111, pp. 441–447, Jun. 2017.

[13] K.-J. Cho, Y.-C. Lee, K. Han, J. Choi, and S.-W. Kim, "No, that's not my feedback: TV show recommendation using watchable interval," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Macao, China, Apr. 2019, pp. 316–327.

[14] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proc. 8th IEEE Int. Conf. Data Mining*, Pisa, Italy, Dec. 2008, pp. 263–272.

[15] Y. Park, J. Oh, and H. Yu, "RecTime: Real-time recommender system for online broadcasting," *Inf. Sci.*, vols. 409–410, pp. 1–16, Apr. 2017.

[16] R. Bambini, P. Cremonesi, and R. Turrin, "A recommender system for an IPTV service provider: A real large-scale production environment," in *Recommender Systems Handbook*. Berlin, Germany: Springer, 2011, pp. 299–331. [Online]. Available: http://www.springerlink.com

[17] L. Nixon, K. Ciesielski, and B. Philipp, "AI for audience prediction and profiling to power innovative TV content recommendation services," in *Proc. 1st Int. Workshop AI Smart TV Content Prod., Access Del.*, Nice, France, 2019, pp. 42–48.

[18] G. Hölbling, A. Thalhammer, and H. Kosch, "Content-based tag generation to enable a tag-based collaborative tv-recommendation system," in *Proc. 8th Int. Interact. Conf. Interact. TV Video*, Tampere, Finland, 2010, pp. 273–282.

[19] H. Li, H. Xia, Y. Kang, and M. N. Uddin, ''IPTV program recommendation based on combination strategies,'' in *Proc. MATEC Web Conf.*, Oradea, Romania, 2018, Art. no. 01003.

[20] S. H. Hsu, M. Wen, H. Lin, C. Lee, and C. Lee, ''AIMED—A personalized TV recommendation system,'' in *Proc. Eur. Conf. Interact. TV*, New York, NY, USA, 2007, pp. 166–174.

[21] N. Chang, M. Irvan, and T. Terano, ''A TV program recommender framework,'' *Procedia Comput. Sci.*, vol. 22, pp. 561–570, Sep. 2013.

[22] Y. Zhang, W. Chen, and Z. Yin, ''Collaborative filtering with social regularization for TV program recommendation,'' *Knowl.-Based Syst.*, vol. 54, pp. 310–317, Dec. 2013.

[23] C. Yu, H. Ding, H. Cao, Y. Liu, and C. Yang, ''Follow me: Personalized IPTV channel switching guide,'' in *Proc. 8th ACM Multimedia Syst. Conf.*, New York, NY, USA, Jun. 2017, pp. 147–157.

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, ''Dropout: A simple way to prevent neural networks from overfitting,'' *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jun. 2014.

[25] Z. Wu, Y. Zhou, D. Wu, M. Chen, and Y. Xu, ''TAMF: Towards personalized time-aware recommendation for over-the-top videos,'' in *Proc. 29th ACM Workshop Netw. Oper. Syst. Support Digit. Audio Video*, Amherst, MA, USA, 2019, pp. 43–48.

[26] T.-W. Yang, W.-Y. Shih, J.-L. Huang, W.-C. Ting, and P.-C. Liu, ''A hybrid preference-aware recommendation algorithm for live streaming channels,'' in *Proc. Conf. Technol. Appl. Artif. Intell.*, Taipei, Taiwan, Dec. 2013, pp. 188–193.

[27] H. Bahn and Y. Baek, ''An intelligent channel navigation scheme for DTV channel selectors,'' *IEEE Trans. Consum. Electron.*, vol. 54, no. 3, pp. 1098–1102, Aug. 2008.

[28] C.-Y. Lin and H.-S. Chen, ''Personalized channel recommendation on live streaming platforms,'' *Multimedia Tools Appl.*, vol. 78, no. 2, pp. 1999–2015, Jan. 2019.

[29] N.-R. Kim, S. Oh, and J.-H. Lee, ''A television recommender system learning a User's time-aware watching patterns using quadratic programming,'' *Appl. Sci.*, vol. 8, no. 8, p. 1323, Aug. 2018.

[30] D. Véras, T. Prota, A. Bispo, R. Prudêncio, and C. Ferraz, ''A literature review of recommender systems in the television domain,'' *Expert Syst. Appl.*, vol. 42, no. 22, pp. 9046–9076, Jun. 2015.

[31] J.-C. Kim and K.-Y. Chung, ''Knowledge expansion of metadata using script mining analysis in multimedia recommendation,'' *Multimedia Tools Appl.*, to be published, doi: 10.1007/s11042-020-08774-0

[32] B. Philipp, K. Ciesielski, and L. Nixon, ''Automatically adapting and publishing TV content for increased effectiveness and efficiency,'' in *Proc. 1st Int. Workshop AI Smart TV Content Prod., Access Del.*, Nice, France, 2019, pp. 51–52.

[33] Y.-D. Seo, E. Lee, and Y.-G. Kim, ''Video on demand recommender system for Internet protocol television service based on explicit information fusion,'' *Expert Syst. Appl.*, vol. 143, Apr. 2020, Art. no. 113045.

[34] M. S. Kristoffersen, S. E. Shepstone, and Z.-H. Tan, ''The importance of context when recommending TV content: Dataset and algorithms,'' *IEEE Trans. Multimedia*, vol. 22, no. 6, pp. 1531–1541, Jun. 2020, doi: 10.1109/TMM.2019.2944214.

[35] J. Oh, S. Kim, J. Kim, and H. Yu, ''When to recommend: A new issue on TV show recommendation,'' *Inf. Sci.*, vol. 280, pp. 261–274, May 2014.

[36] K. He, X. Zhang, S. Ren, and J. Sun, ''Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification,'' in *Proc. IEEE Int. Conf. Comput. Vis.*, Santiago, Chile, Dec. 2015, pp. 1026–1034.

[37] C. Yang, S. Ren, Y. Liu, H. Cao, Q. Yuan, and G. Han, ''Personalized channel recommendation deep learning from a switch sequence,'' *IEEE Access*, vol. 6, pp. 50824–50838, 2018.

[38] J. Wang, P. Huang, H. Zhao, Z. Zhang, B. Zhao, and D. L. Lee, ''Billion-scale commodity embedding for E-commerce recommendation in Alibaba,'' in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, London, U.K., Jul. 2018, pp. 839–848.

[39] K. Zhao, Y. Li, Z. Shuai, and C. Yang, ''Learning and transferring IDs representation in E-commerce,'' in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, London, U.K., Jul. 2018, pp. 1031–1039.

[40] M. Grbovic and H. Cheng, ''Real-time personalization using embeddings for search ranking at airbnb,'' in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, London, U.K., Jul. 2018, pp. 311–320.

[41] S. Oramas, O. Nieto, M. Sordo, and X. Serra, ''A deep multimodal approach for cold-start music recommendation,'' in *Proc. 2nd Workshop Deep Learn. for Recommender Syst.*, Como, Italy, 2017, pp. 32–37.

[42] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, ''BPR: Bayesian personalized ranking from implicit feedback,'' in *Proc. Conf. Uncertainty Art. Intell.*, Montreal, QC, Canada, 2009, pp. 452–461.

[43] A. V. Wyk. Encoding Cyclical Features for Deep Learning. South Africa. [Online]. Available: https://www.kaggle.com/avanwyk/encoding-cyclical-features-for-deep-learning

[44] C. Guo and F. Berkhahn, ''Entity embeddings of categorical variables,'' 2016, *arXiv:1604.06737*. [Online]. Available: http://arxiv.org/abs/1604.06737

[45] R. Zhang, Y. Deng, and L. Shi, ''User research and design for live TV UX in China,'' in *Proc. Adjunct Publication ACM Int. Conf. Interact. Exper. TV Online Video*, Hilversum, The Netherlands, 2017, pp. 9–14.

[46] S. Zhang, L. Yao, A. Sun, and Y. Tay, ''Deep learning based recommender system: A survey and new perspectives,'' *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–38, Feb. 2019.

[47] R. Devooght and H. Bersini, ''Long and short-term recommendations with recurrent neural networks,'' in *Proc. 25th Conf. User Modeling, Adaptation Pers.*, Bratislava, Slovakia, Jul. 2017, pp. 13–21.

[48] M. He, Q. Meng, and S. Zhang, ''Collaborative additional variational autoencoder for top-N recommender systems,'' *IEEE Access*, vol. 7, pp. 5707–5713, 2019.

[49] X. Glorot and Y. Bengio, ''Understanding the difficulty of training deep feedforward neural networks,'' in *Proc. Int. Conf. Art. Intelligen. Statis.*, Sardinia, Italy, 2010, pp. 249–256.

XIAOSONG ZHU is currently pursuing the Ph.D. degree with the College of Information Science and Engineering, Yanshan University, Qinhuangdao, China. His current research interests include machine learning, data mining, and recommender systems.

JINGFENG GUO is currently a Professor with the Faculty of Computer science, Information Science and Engineering, Yanshan University, Qinhuangdao, China. He has authored over 130 journals and conference papers, since the inception of his research career. His main research interests include social network, database theory and application, and data mining.

SHUANG LI is currently pursuing the Ph.D. degree with the School of Architecture, Tianjin University, Tianjin, China. She is also a Vice Professor with the Faculty of Ecology, Hebei University of Environmental Engineering (Environmental Management College of China), Qinhuangdao, China. Her current research interests include multimedia and digital landscape.

TONG HAO is currently pursuing the master's degree with the College of Information Science and Engineering, Yanshan University, Qinhuangdao, China. Her current research interests include social network analysis, machine learning, and deep learning.