IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received July 28, 2018, accepted August 29, 2018, date of publication September 10, 2018, date of current version October 8, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2869470

Personalized Channel Recommendation Deep Learning From a Switch Sequence

CAN YANG[1], (Member, IEEE), SIXUAN REN[1], YONG LIU[2], (Fellow, IEEE), HOUWEI CAO[3], QIHU YUAN[1], AND GUOQIANG HAN[1]
[1]College of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China
[2]Department of Electrical and Computer Engineering, New York University, Brooklyn, NY 11201, USA
[3]Department of Computer Science, New York Institute of Technology, New York, NY 10023, USA

Corresponding author: Can Yang (cscyang@scut.edu.cn)

This work was supported in part by the National Natural Science Foundation of China (NSFC), under Grant U1611461, in part by the Science and Technology Plan Project of Guangdong, China, under Grant 2014B010115002, and in part by the Guangzhou Science and Technology Program Key Project, China, under Grant 201704030124.

ABSTRACT Internet protocol TV (IPTV) services could enhance personalized viewing experience in a more interactive way than traditional broadcast TV systems, but it is still difficult for subscribers to quickly find interesting channels to watch from a huge selection. This paper focuses on a framework for personalized live channel recommending via deep learning from a historical switching sequence with a long short-term memory (LSTM) neural network. Using real-world IPTV watching logs, we first obtained insights into user behaviors when watching live channels, and then proposed a learning scheme on how to dynamically generate a recommended channel list for each user with an independent LSTM net trained using the channel watching history during a slide window. For designing a good data architecture and representation scheme for a dynamically learning framework, we then studied the performance of the proposed recommendation method by varying the width of the slide window for training, the length of input sequence for prediction, and the mode to process input and label space. We finally developed a separate learning method to fairly recommend for popular (hot) or unpopular (cold) channels, respectively, based on channel popularity in the training set with an extra price of a possible hit lag after recommendation, in order to alleviate the Matthew effect arising from the conventional recommendation based on historical information. The experimental results show LSTM succeeds in learning from a historical channel switching sequence, outperforms several baseline recommendation methods, especially for hot channels, and the classified recommendation by separate learning brings an overall performance gain.

INDEX TERMS Deep learning, IPTV, long-short term memory, recommender systems, recurrent neural networks, separate learning, user behavior analysis.

I. INTRODUCTION

Internet Protocol TV (IPTV) is a significant advancement in the TV industry and also brings some new challenges to viewers. An IPTV platform usually provides so many live channels that it becomes difficult for a viewer to quickly find the next interesting channel to watch whenever he wants to switch channel. The viewer has to keep zapping until reaching an interesting one. Long channel switching response time seriously degrades the user Quality-of-Experience (QoE). Meanwhile, frequent switches between unwanted channels consume excessive resources at both server and client sides. It is therefore of necessity and significance for IPTV systems to mine the potential interests of viewers and guide them to quickly find interesting channels to watch. We design a recommender system to dynamically generate personalized Electronic Program Guide (EPG) to achieve this goal. Recommendation Systems (RS) can be used to filter out some unwanted channels. Meanwhile, the conventional recommenders based on the previous choices probably lead to the so called "Matthew effect". Unpopular channels have less chance to be presented to viewers to gain popularity. Therefore, we also intend to develop a separate learning method for popular and unpopular channels and recommend them fairly.

IPTV offers both live TV broadcasting and video on demand (VOD), but there is a major difference between them.

Popular VOD services, such as Netflix and YouTube, can provide video recommendations for their users. In this paper, we focus on recommendation of live TV channels. Unlike VOD content, a live TV channel consists of a succession of programs. A live streaming server streams channel programs following a pre-arranged schedule in the form of EPG. However, it is inconvenient for users to browse through a long EPG to find the next channel to watch. Instead, they just keep zapping through channels until hitting an interesting one. RS of live channels generates a personalized short channel list for a user to consider. If the generated recommendation list matches the user's interest well, it will greatly reduce the channel zapping time and alleviate resource consumption in the IPTV system. Besides EPG recommended list, accurate channel prediction can guide channel pre-loading for fast channel switching. For example, Yang and Liu [1] introduced some relevant contents of channel switching in IP-based TV systems in detail. However, from the recommendation point of view, the characteristics of live channels are more sophisticated than VOD contents. First, the most significant character is the time-awareness of live channels. All programs are delivered with a pre-arranged schedule. If a viewer wants to watch a specific program, he has to switch to a channel at a specific time when the program is broadcasted on that channel. On the contrary, VOD systems let a viewer watch any content at anytime. And then, the watch duration of live channels has more diverged distribution than VOD contents with certain expectation of playback length. Thirdly, a user often keeps zapping and generates lots of browsing records within a short duration until he finds an interesting channel. Those channel zapping records do not reveal real user interests and have to be filtered out as noise.

Focusing on a historical channel switching sequence in IPTV logs, we propose a Personalized Channel Recommendation System, called PCRS, based on a Long Short Term Memory (LSTM) neural network [2]–[4], which is a classical Recurrent Neural Network (RNN). Only by learning from one-dimension vectors consisting of channel switching sequences, the PCRS works very well and gains good recommendation performance. Although it is helpful for a conventional recommendation system to fuse some channel/TV program meta data and descriptions, user profiles, and social connections among them, this paper focuses on studying the impact of the channel switching sequence adopted in the PCRS. Our experimental results demonstrate that LSTM significantly outperforms the other baselines and our prior methods [5] by only using the channel sequence vector. In PCRS, we design a sliding window to dynamically generate recommended channels list while PCRS implicitly processes varying channel labels in real time series to enhance the efficiency of the LSTM algorithm. The LSTM of each user is trained with the recent channel watching activities in the current sliding window. Then, it is used to predict which channels the user is likely to watch the next. For comparison, we also introduce several baseline channel RS models. We compare their performance using experiments driven by real IPTV user channel watching logs provided by a middle-scale IPTV provider in southern China.

In fact, a majority of the current recommendation algorithms, reviewed in Sec. II, are driven by historical information. They probably lead to the so-called Matthew effect [6], in which unpopular channels will seldom be presented to viewers to gain popularity. In recent, from a probabilistic viewpoint, Rocío and Pablo analyzed the effectiveness of popularity in recommender systems [45]. To alleviate this Matthew effect, we further propose a separate learning (SL) approach for recommending TV channels with different popularity. That is, we recommend hot channels for the user with an artificial neural network (ANN) trained with previous popular channels in watching logs (called HANN). Likewise, we recommend cold channels for the user with another ANN network trained with the unpopular channels in the same period (called CANN). Here, we consider a channel being hot if its popularity is more than a given threshold (e.g. $> 4\%$) in the training set, otherwise as cold. Moreover, some hot channels of a viewer are maybe cold referring to another viewer, vice versa, because one channel has probably different popularity in logs for different viewers. By giving fair recommendation chance for popular and unpopular channels, the SL approach is able to mitigate the Matthew effect, even though not completely wipe it out. Our main contributions are summarized as follows.

1) We presented a framework of personalized channel realtime recommendation systems (PCRS) using LSTM to learn from a channel switching sequence for live channel recommendation of each user.

2) Through extensive experiments on real IPTV user behavior logs, we found some fundamental characteristics on the channel recommendation system's performance and insight into the impact from the width of sliding training window, the length of input sequence for LSTM, the data organization of Artificial Neural Networks (ANN).

3) We found that the proposed PCRS is well suitable for recommending popular/hot channels while we presented a separate learning method to improve the recommendation performance by fairly processing unpopular/cold channels and popular/hot ones.

The rest of this paper is organized as follows. Sec.II discusses the related work, and Sec.III focuses IPTV user behaviors analyses. We propose the framework of PCRS, and the LSTM model, baselines and evaluation methods associated with PCRS in Sec.IV. Furthermore, Sec.V presents data representation and organization for PCRS. The experimental results and analyses are reported in Sec.VI. We also make a discussion in Sec.VII, and draw a conclusion in Sec.VIII.

II. RELATED WORK

This work focuses on a LSTM-based TV live channel recommendation system. In this section, we briefly go through the related work on recommender system, recommendation on TV programs and live TV channels, and the applications of LSTM neural networks in related research.

A. RECOMMENDER SYSTEM

In nature, live channel recommendation is a branch of recommender systems, which studies the patterns of user behaviors and suggests items of potential interests of users. A literature review of recommender system [7] introduced a variety of RSs. Nowadays, RS has been widely applied in news [8], music [9], movies [10], health [11], [12], telecom products [13], electronic commerce [14] and so on. Zhang *et al.* [15] utilized real-time micro-blogging information to recommend news for target users. Briguez *et al.* [16] proposed a novel movie RS for a family, which used Defeasible Logic Programming (DeLP) to decide whether a movie suggestion should be sent to a user. Social network based top-K recommender was studied in [17]. Collaborative filtering is widely used in recommender systems and can be combined with other methods to improve recommendation accuracy. For example, Ling *et al.* [18] combined content-based filtering (CB) and collaborative filtering (CF) to generate recommendation. Graph-based recommendation also absorbs some research interest [19]. Zhang [20] presented a novel group recommender system, and the diversity of group recommendation was investigated in [21]. Although cold boot and group diversity are the research hotspot of RS at present, few people study on classified learning by popularity in the training set. We particularly focus on how to provide a fair chance for unpopular contents.

B. RECOMMENDING TV PROGRAMS AND CHANNELS

We consider TV program recommendation as the most related work to live channel recommendation because of the same TV application background. The main difference between recommending live channels and TV programs and others is the time-effectiveness. For an instance, a video in a VOD repository can be selected by viewers at anytime after being recommended. TV programs in live channels are broadcasted according to some fixed schedule. Recommendations ought to be generated to match the program schedule to be effective. Therefore, the goal of live channel recommendation is to match the interest of a viewer when he/she wants to watch it. However, in the past, more attention has been paid to program content in TV program recommendation. We treat a TV channel as a time series comprising of TV programs. Channel recommendation is to serve for those interested in a channel but not for a detail program of the channel. We need to take more factors into account in channel recommendation than TV program recommendation, but some of recommendation methods on TV programs can also be used in channels recommendation.

1) TV PROGRAM RECOMMENDATION

Recommendation of individual TV programs broadcasted on each channel has been studied in recent years. Ras *et al.* [22] gave a nice survey on TV program recommendation. Many studies on TV program recommendation are based on collaborative filtering [15], [23], [24]. Kim *et al.* [23] proposed a new algorithm that clusters TV users and recommends TV programs without using user program ratings. Zhang *et al.* [15] combined TV program recommendation with social network information using probabilistic matrix factorization and achieved good performance. With the current tide of machine learning, ANN has been adopted for TV program recommendation [25], [26]. Pyo *et al.* [27] used LDA to recommend TV programs, Kristic and Bjelica [25] used Resilient Back-Propagation and Extreme Learning Machine to predict the type of TV programs users like. In fact, a TV program is broadcasted on a TV channel according to some schedule, and it is a great challenge to recommend a suitable program for a viewer when she/he happens to switch to the channel. Similar to live streaming, a channel recommender, instead of a program recommender, is more suitable for live broadcast.

2) LIVE CHANNEL RECOMMENDATION

Live channel recommendation is important for IPTV services, but has received less attention than TV program recommendation. Unlike a TV program recommender, the channel RS works in realtime to recommend to a user the next channel to watch. Zui *et al.* [28] provided a hybrid preference-aware recommendation algorithm. Ning *et al.* [29] discussed the difficulty of live channel recommendation and proposed ways to intelligently recommend channels to users. Oh *et al.* [30] studied when to recommend TV channels to users without annoying them. Bahn and Baek [31] analyzed user channel watching behaviors to recommend channels. Zhu *et al.* [32] presented a personalized approach based on label of contents in 2014. In [5], we utilized a subset of real IPTV watching logs and integrated a variety of methods to predict next channel.

C. LSTM AND OUR WORK

Different from the related work, this paper focuses on a personalized live channel recommendation using LSTM to learn from a historical switching sequence (channel IDs) of each user, by which the results of this paper outperform those using a fusion of conventional channel recommenders in our prior work [5]. LSTM is a well-known recurrent neural network (RNN) [2], [3] and the academic community shows strong interest in deep learning by LSTM [4] in recent years, especially in Natural Language Processing (NLP) [33], Semantic Video Segmentation [34], financial data analysis [35], and so on. We seldom found LSTM to be used for TV channel recommendation. The similar work to our research is related to time series process with modified GRU in film and music recommendations [36], and modified LSTM in music and literature citation recommendations [37]. We furthermore researched on a separated RS model for hot and cold channels, which is not found in previous work.

III. USER BEHAVIOR ANALYSIS

For precisely designing the data architecture of an IPTV RS, we first focus on analyzing user behaviors of watching live

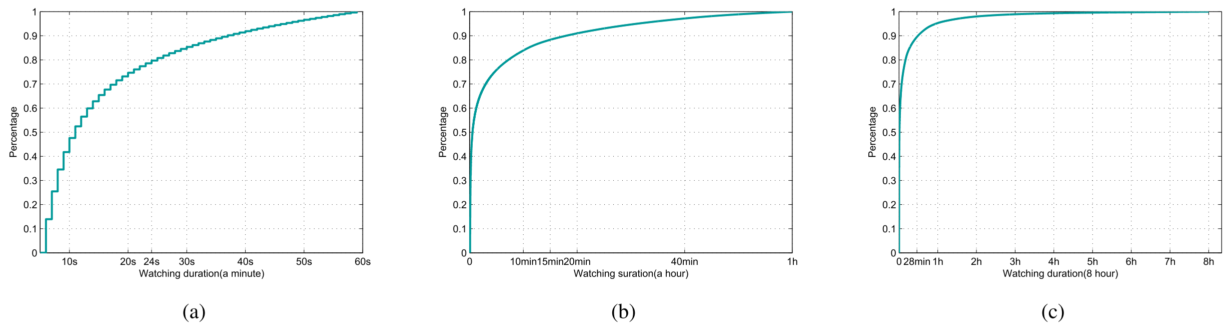


FIGURE 1. **CDF of watching duration (τ) in three ranges. (a) τ <60s. (b) τ <1h. (c) τ <8h.**

channels. An IPTV server can collect the watching history of users' Set-Top-Box (STB) and store them into a database. A user typically visits IPTV channels in the following ways: 1) **S**tart a channel watching session; 2) **W**atch the channel; 3) **C**hange into the next channel; 4) **T**erminate the current watching session. The traces from the process of **SWCT** generate IPTV logs. A dataset used in this paper is the user channel watching logs of a major IPTV service provider in Southern China. There are totally 220k devices and 156 channels in the original dataset, which includes tons of user channel zapping information and other noises. For training and predicting, we provide 3 fundamental rules for data preprocessing based on statistical measure in the IPTV dataset with the consideration of the duration of channel watching, the gap between watching sessions and the popularity of channels. The 3 rules are listed as follows,

Rule 1: to extract channel watching records with the duration ranging from ten seconds to one hour;

Rule 2: the gap between two adjacent channel switching records should be less than 20s, otherwise it should be to shut down and restart to Watch TV (abbr. SARWT);

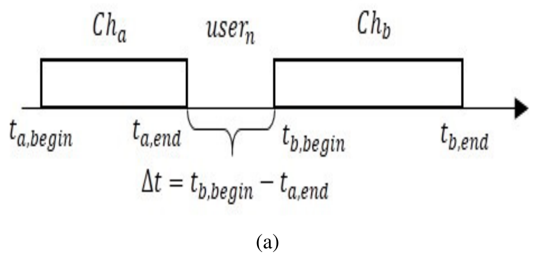
Rule 3: if the number of accesses to a channel by a subscriber is more than the threshold percentage $\hbar$ of total number of channel accesses, the channel is called a hot channel for the subscriber, otherwise the channel is considered cold.

The primary problem of IPTV channel recommendation is to recommend ***hot*** channels, because the cold ones would contribute only a little to A_{cc}. We define an empirical parameter, noted $\hbar$, to distinguish hot and cold channels on the per-user basis and find 4% is a good empirical value of $\hbar$ by sampling investigation on the IPTV dataset. Based on this definition, the histogram of the number of hot channels of a user is shown in Fig.3(d), from which we find only a few popular channels, about 7 to 11 channels, are frequently accessed by a user. However, it would be unfair for cold channels recommendation if we only take the popularity into account, the recommendation based on distribution of channels choices would furthermore lead to a critical Matthew effect [6], [38]. To alleviate this phenomenon, we presented a classified recommendation based on the popularity with a fair chance for specially recommending cold channels.

In the following experiments, we extract channel sequence data according to Rule1 and Rule 2 from the original data set, and form the switching channel chain for training and testing. Rule 3 will be used for enhancing learning performance. In detail, the 3 rules originate from statistics analysis of this dataset shown in Sec. III-A, Sec. III-B, and Sec. III-C. As for a sample of the watching logs, each entry consists of four fields: *{Device_id, Channel_Number, Start_Time, Duration}*. Here, *Device_ID* denotes a subscriber, i.e., a user or a family; *Channel_Number* represents the channel the user is watching; *Start_Time* is when the user starts watching this channel; and *Duration* is how long the user stays in the channel.

A. DURATION OF LIVE CHANNEL WATCHING

The main goal of live channel RS is to recommend the next interesting channel for a user to watch whenever he/she is done with watching a channel. To mine the actual interest of a user, we have to filter out zapping channels and channels not actively watched. Since the user logs do not directly tag user channel zapping and leaving events, we have to infer these events by examining the channel watching duration. As for the IPTV dataset, the cumulative distribution function (CDF) of watching duration shorter than one minute is shown in Fig.1 (a), from which we found about 45% of them are shorter than 10 seconds. This suggests that a mass of zapping events for finding interesting channels. Fig.1(b) shows the CDF of watching duration less than one hour, in which over 83% of them are shorter than twenty minutes. Fig. 1 (c) is the CDF of all duration, in which more than 94% of user watching duration is less than one hour. That is, $0.94 * 0.83$ (over three-fourth) of the all the durations are less than 10 minutes. In the prior work [5], we paid much attention to long viewing duration behavior in range of [10 *minutes*, 5 *hours*]. However, the above statistics show that a majority of viewing durations are less than 10 minutes. We thus extend our duration range to cover more view durations in this paper. In addition, durations shorter than 10 seconds are considered as zapping and should be filtered out. Similar statistics have been obtained in prior work [39], [40]. Consequently, to mine user watching interest and generate meaningful recommendation, our PCRS only focuses on channel watching records with duration in the range of [10*seconds*, 1*hour*].



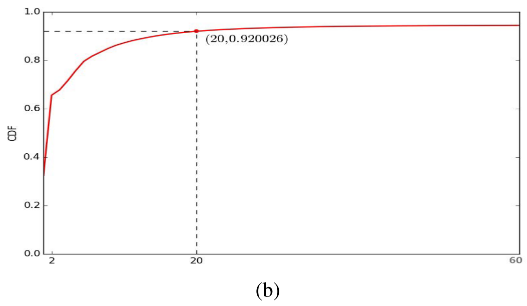


FIGURE 2. Switch behavior and the gaps of two records. (a) A case of switch behavior. (b) CDF of the gaps Δt.

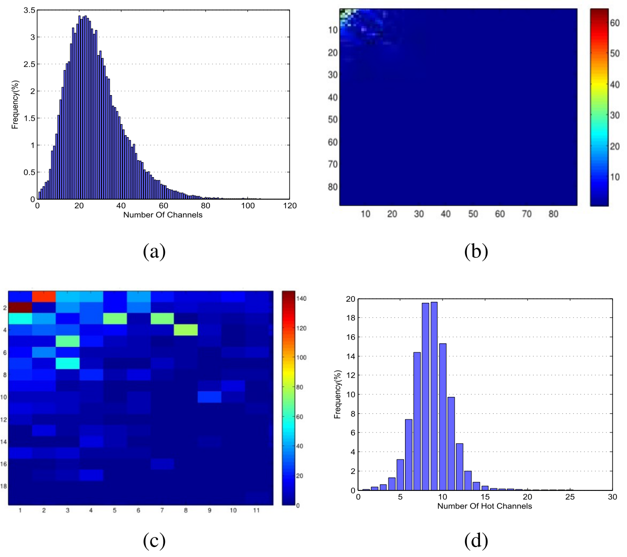


FIGURE 3. Statistics of Channel Visits and Switches. (a) Number of visited channels. (b) Switch matrix of the channels. (c) Switch matrix of top 20 channels. (d) Number of visited hot channels.

Empirically, we propose **Rule 1** for data pre-processing. Note that the duration is only used for selecting effective viewing or switching records and not acted as an input for ANN training in PCRS.

B. GAP BETWEEN WATCHING SESSIONS

How to identify boundary between two watching sessions is a critical issue. A STB produces a record for each channel watching event by a user and sends it to an IPTV database. There is no explicit record about when the user turns on/off his TV to start/end a watching session. Two adjacent records produced by a subscriber's STB are possibly derived from two cases. One is switching from one channel to another within the same watching session, the other is to Shut down and Restart to Watch TV (abbr. SARWT), i.e., the two records belong to two different sessions. As shown in Fig.2, $user_n$ starts to watch ch_b at the time $t_{b,begin}$ after he finishes watching Ch_a at $t_{a,end}$. The gap between the two records is $\Delta t = t_{b,begin} - t_{a,end}$. We use Δt to infer the boundary between two sessions, that is, a SARWT event occurs if $\Delta t > Th$. The CDF of intervals between two adjacent records is shown in Fig. 2(b). To determine the value of Th, we notice that about 66.71% of records on Δt are less than 2 seconds, 79.16% less than 5 seconds, and 92% less than 20 seconds. Thus, we define a discrimination rule to detect boundary between sessions: if the interval of two adjacent records is less than 20s, we consider the user switches from one channel to another within the same session; if Δt more than 20s, we assume the user turns off TV first and restarts another session later (SARWT). Different from conventional channel switching, SARWT events actually are noise for RS and should be marked out and processed in special way when we construct a training sequence. Thus, we present **Rule 2**.

C. POPULARITY OF CHANNELS

In the original logs, there are 156 channels, the popularity of each channel is different among different user populations. We plot a histogram of how many channels visited by a typical user during one week as shown in Fig.3 (a). The numbers of visited channels of most users range from 6 to 50. To gain an insight into the characteristic of switching between channels, we model the behavior of channel switching as a $n \times n$ square matrix M_{nn} if there are n channels in an IPTV system. To make a better presentation of switching process, for each subscriber, we reorder the channel IDs in descent according to their popularity for the subscriber. The hotter the channel, the smaller the channel ID. That is, a less accessed channel has a higher ID. We set matrix entry $M_{nn}(i, j)$ as the count of switching from Channel i into Channel j. We visualize the switching matrix of the typical subscriber in Fig.3(b) and Fig.3(c), in which the depth of color at (i, j) represents the frequency of switches from channel i to j. It's obvious in Fig.3(b) that the channel switches are concentrated in the upper left corner. To see more clearly, we zoom in to focus on channel switching to/from the top 20 channels in Fig.3(c). It is clearly shown that the vast majority of channel switches occur between the top 10 popular channels, which is similar to the work on movies of Bjelica [42]. We use the empirical parameter $\hbar$ to distinguish hot and cold channels on the per-user basis and give out the aforesaid **Rule 3**.

IV. PERSONALIZED LIVE CHANNEL RECOMMENDER

After user behavior analyses, we first propose the framework of PCRS, and then introduce the adopted LSTM model and several relevant problems.

The PCRS utilizes LSTM neural networks for IPTV real-time live channel recommendation whenever a user wants

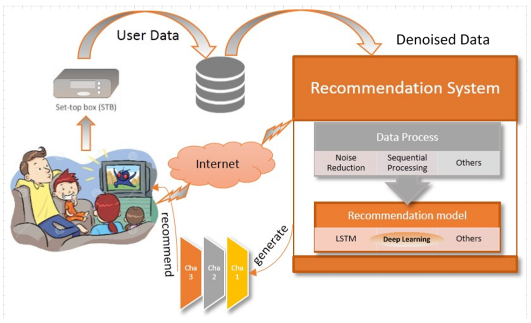
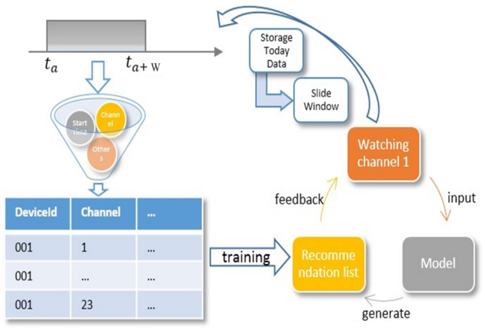


FIGURE 4. The framework of PCRS.



DeviceId	Channel	...
001	1	...
001	...	...
001	23	...

FIGURE 5. The work principle of PCRS.

to switch channel, and the weights of the neural networks would be updated every day so as to be adapted to the evolution of user behaviors and channel characteristics. Our proposed methods based LSTM neural networks can work not only in client but also in server. Fig. 4 illustrates the overall framework of server-based PCRS. In an IPTV system, a user's TV watching logs are collected from the STB, a Customer Premise Equipment (CPE), and then sent to a central database at the IPTV server side. PCRS uses the collected logs to train an ANN and finally pushes a recommended channel list to the STB of each target user. As for OTT-based TV applications, the clients are connected to the central server through Internet. PCRS running on the server will dynamically generate a personalized EPG, which will be shown on the screen of the target client in forms of fading-in and fading-out, floating or barrage, etc. Also, a client-based PCRS can learn with its own watching TV logs and conduct an individual recommendation with no need of data from other viewers' STBs.

In PCRS, an ANN is trained with user channel watching logs in a sliding window, the width of which is a tunable parameter denoted as W. Fig. 5 illustrates the work principle of PCRS with the following steps for recommending channels to a target user.

In this paper, we customize LSTM for IPTV channel recommending. Note that, the channel sequence of switching can be considered to be equivalent with that of viewing, after zapping noise and SARWT are filtered.

A. LONG SHORT-TERM MEMORY IN THE MODEL

It is well known that LSTM has an innate ability of representing time series. In IPTV, the channels watched by a user within a period naturally form a time series. Inspired by NLP [33], we use the channel switching sequences of a user as input vectors to generate an output vector that indicates the next channel the user wants to watch. The skeleton of LSTM sequence is shown in Fig.6(a), and the internal detail of a LSTM cell is shown in Fig.6(b). In Fig.6(a), $x_t, \cdots, x_{t+n}$ are the most recent $n+1$ channels watched by the user, y_{t+n} is the next channel he/she is likely to switch into. Here, n denotes the location of the node in the sequence. As illustrated in Fig.6(c), we customized the LSTM model referring to [35] and [41] and formulated it as follows:

$$\begin{cases} i_t = sigmoid(W_{ih}h_{t-1} + W_{ix}x_t + b_i) \\ o_t = sigmoid(W_{oh}h_{t-1} + W_{ox}x_t + b_o) \\ f_t = sigmoid(W_{fh}h_{t-1} + W_{fx}x_t + b_f) \\ g_t = tanh(W_{gh}h_{t-1} + W_{gx}x_t + b_g) \\ c_t = g_t \odot i_t + f_t \odot c_{t-1} \\ h_t = o_t \odot tanh(c_t) \\ \hat{y}_t = softmax(h_t \odot W_{hy} + b_y) \end{cases} \tag{1}$$

Here, i_t, o_t, f_t denote input, output and forget gates at time t, respectively. And g_t denotes the distorted input to the memory cell at time t, c_t denotes the content of memory cell, h_t denotes the value of hidden node, the symbol $\odot$ denotes an element-wise product operation. For LSTM training, $\hat{y}_t$ represents the predicted likelihood of output channels with a softmax function, y_t^* denotes the vector [43] of input labels transformed from the original channel number, and $\hat{y}_t = softmax(h_t \odot W_{hy} + b_y)$, here b_y denotes a bias. When training, the weights (W_{hy}) are adjusted with stochastic gradient descent (SGD) while the loss function is based on a given distance between y_t^* and $\hat{y}_t$. We also illustrate the connections between the hidden layer and the output layer in Fig. 6(b), and the internal structure of a LSTM cell in Fig. 6(c) referring to [41]. As shown in Fig.6(b), both input nodes and output nodes have the same vector length M, namely the number of channels in a IPTV system, e.g., $M = 156$ in our dataset. And Q represents the number of hidden nodes and is set to be 30 for each unit of the LSTM in the latter experiments. After training the ANN via multiple iterations, we obtain the output likelihood vector $\hat{y}_{t+n}$, and then compute $argtopK(\hat{y}_t^j)$ of candidate channels, and finally form a recommended list.

B. BASELINE RECOMMENDERS

To compare with LSTM, we introduce several baseline Top-K recommenders, called PS (i.e. Personal Schedule in [5]), PP (i.e. Personal Popularity in [5]), collaborative filtering by Singular Value Decomposition (SVD) in [24] and a proposed SVD PW (Pairwise), respectively.

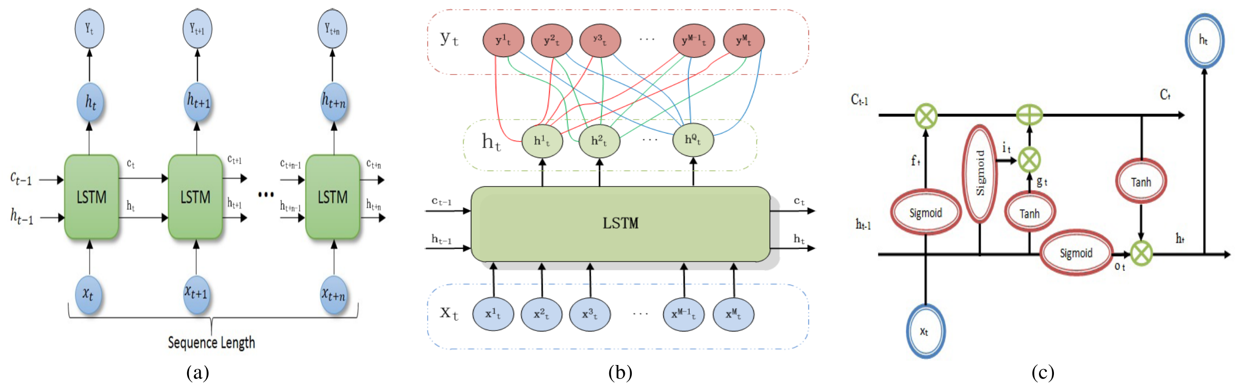


FIGURE 6. The hierarchical structure of LSTM. (a) The skeleton of LSTM sequence. (b) LSTM hidden layer and output layer. (c) Internal structure of a LSTM cell.

1) PS

In order to generate Top-K recommendation list for a target user u at time t, we first find the time slot to which t belongs. In the experiments on PS, we divide each hour into four time slots, and each slot lasts for 15 minutes. We then calculate how many times user u accessed each channel in the same time slot of every day in the sliding window. All the channels user u watched are ranked in descent by their accessed frequencies, and the Top-K channels in the ranked list will be recommended to the user. For example, if a target user u wants to switch channel at $9:23$ am, and the sliding window is W (e.g. 7 days), the top K channels that u watched the most from $9:15$ a.m. to $9:30$ a.m. every day in the previous week will be recommended to user u.

2) PP

Unlike PS, all the channels are ranked based on the number of accesses by the target user in the whole sliding window, regardless of the time slot in the day. Like PS, the top K channels that user u watched the most in the previous week will be recommended to user u.

3) SVD

As for SVD used in channel recommendation systems, we construct a score matrix $\mathbb{S}_c$, in which the row means user ID, the column stands for channel ID, and the value of cell of the matrix ($\mathbb{S}_c(i,j)$) is the probability of user i switching into channel j.

4) SVD PW

Under the framework of CF with general SVD, we further propose another SVD method based on the pairwise channel switching probability, called SVD PW, by which we build a switching score matrix with the probability of the pairs of two adjacent channels watched by one user.

More details have already been used for studying channels recommendation on the dataset and relevant results were shown by Yu *et al.* [5].

C. DEFINITION OF EVALUATION METRICS

For the performance evaluation, we defined an accuracy (noted A_{cc}) as following formula in the work.

$$A_{cc}(u,K,d) = \frac{\mathbb{I}\,(RealCh_i(u) \in R_i(u,K))}{N_u^d} \tag{2}$$

$$A_{cc}(K) = \frac{1}{U*D}\sum_{u=1}^{U}\sum_{d=1}^{D} A_{cc}(u,K,d) \tag{3}$$

Here, $A_{cc}(u,K,d)$ represents the Top-K recommendation hit ratio of the user u on day d, $A_{cc}(K)$ denotes that of all the users during D days, N_u^d denotes the total recommending times of user u on day d, U is the total number of users, $R_i(u,K)$ is the set of the recommended K channels to user u before the i-th channel switch, $RealCh_i(u)$ is the real channel watched by user u after the i-th recommendation event, and $\mathbb{I}(\cdot)$ represents the indicator function. Based on the aforementioned workflow model of PCRS and the illustration of Fig. 5, the recommending events will occur step-by-step when watching TV channels, so the precision can be considered to be equal to the recall because the times to recommend are equal to the times to watch in the recommendation system. For avoiding confusion on meaning of precision and recall, we adopted a unified metric of A_{cc} defined as Eqt.(2) and Eqt.(3) if there is no particular declaration in the paper.

V. DATA ORGANIZATION AND REPRESENTATION

Data organization and representation in the LSTM architecture have significant impacts on the performance. We design an independent LSTM neural network in the PCRS for each viewer to learn from his/her historical channel switching sequences, involving in data representation, sequence construction, sliding window, channel label compression, randomization of training data, and popularity-based channel classification.

A. DATA ORGANIZATION FOR TRAINING

1) HOW TO SELECT USERS?

Based on the statistical analysis in Sec.III, we need to extract useful data from the original dataset and then load them into our model. The watching duration in the original dataset ranges from 1 second to several hours. According to **Rule 1** in Sec.III, the record whose duration is outside of [10seconds, 1hour] should be filtered out. To provide enough training data for the model, we chose the users who switched channels more than 300 times. As a result, 1.62 million records from 3k STBs were finally extracted from the original data from 220k STBs for our experiments.

2) HOW TO CONSTRUCT TRAINING SEQUENCES?

We need to make LSTM work well with a suitable representation of training data. According to [26], the sequentially watched channels of a viewer can be treated as a channel sequence. In spite of the behavior of SARWT, we assume a user turns off TV at the end of a day. The construction of the training sequence is similar to an overlapped slide window. Given the watched channel sequence of a certain viewer on some day, e.g., $\langle x_1, x_2, ..., x_L\rangle$, here L denotes the total length of the switch sequence. How should we convert it to training sequences for LSTM? In our solution, a training matrix X and an output label matrix Y are designed as Eqt.(4) and Eqt. (5), respectively. At first, we can set the length of training sequence (noted L_{ts}), Secondly, we may divide the original sequence into a series of input training sequences with the given length (L_{ts}) as Eqt.(4). Thirdly, the output labels for training are created as in Eqt.(5), which is essentially the input matrix X shifted ahead by one time unit. Except for the first and the last channel in the switch sequence, each channel acts as the input of next channel, as well as the label of the last one in the training matrix. The size of the matrix is $L_{ts} * (L - L_{ts})$.

$$X = \begin{pmatrix} x_1 & x_2 & \cdots & x_{L-L_{ts}} \\ x_2 & x_3 & \cdots & x_{L-L_{ts}+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{L_{ts}} & x_{L_{ts}+1} & \cdots & x_{L-1} \end{pmatrix} \tag{4}$$

$$Y = \begin{pmatrix} x_2 & x_3 & \cdots & x_{L-L_{ts}+1} \\ x_3 & x_4 & \cdots & x_{L-L_{ts}+2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{L_{ts}+1} & x_{L_{ts}+2} & \cdots & x_L \end{pmatrix} \tag{5}$$

Note that, according to **Rule 2**, the first watched channel after a SARWT event cannot act as label, but only as input in training. Similarly, the last watched channel before a SARWT event cannot act as input but only as label. That is, a SARWT event leads to a vacancy of original watching sequence and a training process should jump over the SARWT location. At the same time, to avoid local minimum, we randomly permute the rows of X (Eq.4) and Y (Eq.5) (X and Y using the same permutation order) for training.

3) HOW TO COMPRESS CHANNEL LABEL SPACE?

In the harnessed LSTM model, the data type as polynomial or string cannot be directly applied for computing in neurons. The processing of channel labels become therefore unavoidable and significant in use of LSTM neurons. Here, the label space refers to the range of channel number. The original number of a channel in the dataset acts as the identification and the distance between two channel numbers has no numerical significance. Each user often watched only a few of the channels provided by the TV vendors. Although the number of cold channels is usually more than that of hot channels for a viewer, the number of switches between cold channels is much less than that between hot channels. In our experiments, hot and cold channels are classified by **Rule 3** in Sec.III. Note that the channel popularity of each user is different. Furthermore, we found a compact label space can improve the accuracy and reduce computation costs. Although there are 156 channels in our dataset, the actual number of channels watched by a viewer in a given duration is much less than 156 and varied every day. For all channels watched by each user in each training window, we compress their IDs into a compact set by a hash operation, noted as $\mathbb{M}(i) \xrightarrow{H} \mathbb{M}'(i)$, here *Hash* Function is expressed as $M'(i) = argsort(M(i))$ in Python syntax. For instance, if the original channel set includes four channels as $\{\mathbb{M}=[30\ 128\ 65\ 104]\}$, their IDs are compressed by hash as $\{\mathbb{M}'=[1\ 4\ 2\ 3]\}$. Following the representation of input training data, the output label is represented with one-hot vector in LSTM. This adaptive compression on label space in the PCRS alleviates the deviation of data in order to improve the performance of LSTM neural networks for IPTV recommending. On the effectiveness of label compression is shown in Sec.VI-D.

B. TWO FACTORS ON DATA PROVISION

1) LENGTH OF CHANNEL SEQUENCE

With respect to the recommendation of LSTM, we need to set the length of an input sequence, denoted as parameter L_{ts}, which is also the number of rows in input training matrix X. L_{ts} determines how much information of LSTM can be used for prediction. For an instance, if $L_{ts} = 5$ in LSTM, the last five watched channels of a viewer are used for predicting the next channel he/she is likely watch. In theory, there exists a L_{ts} to achieve the highest accuracy. In our experiments, L_{ts} does affect A_{cc}, and we empirically found a good L_{ts}. Intuitively, the shorter the sequence length, the less information the model can get. Inversely, if one sets L_{ts} too long, the memory of LSTM may be interfered, and the computation cost for training will accordingly increase. More discussion of this trade-off will be addressed in Sec.VI-A.

2) WIDTH OF TRAINING WINDOW

TV programs in IPTV live channels are updated every day by IPTV providers. In general view, the older watching traces might no longer be relevant for the current recommendation, and the newer would influence the prediction. In the PCRS, we define a parameter W as the width of the sliding training window. Fig. 7 illustrates how a recommendation model is trained with the sliding window.

Let t_a be the start time and W the width of a training window, with day as time unit, the PCRS uses the watching logs in $[t_a, t_{a+W}]$ for training, and predicts what a user will prefer at each channel switch in the day of t_{a+W+1}. When

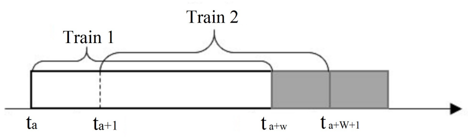


FIGURE 7. How a train window slides for updating train data.

Algorithm 1 Separate Learning With Popularity for a User

```
Begin :                                                % Notes %
 1:  $W, D, M, L_{ts}, \hbar$                           Initializing
 2:  for $d = 1 : D$
 3:     $S_o \in [d, d + W]$;                            Train set
 4:     $Randomize(S_o)$;
 5:     $Compress(Label \in S_o)$;                       [Optional]
 6:     $S_t \in [d + W + 1]$;                           Test set
 7:     $P_m = count(S_o(m))/count(S_o)$;                Channel Popularity
 8:     $S_o^h \leftarrow C_h := FindHot(P_m > \hbar)$;    hot channel records
 9:     $S_o^c \leftarrow C_c := FindCold(P_m \leq \hbar)$;  cold channel records
10:     $HANN \leftarrow Training(S_o^h)$;               Train HANN
11:     $CNHH \leftarrow Training(S_o^c)$;               Train CANN
12:     $S_t^h : m \in C_h|S_t; S_t^c : m \in C_c|S_t$;
13:     $A_h \leftarrow Testing(HANN, m \in S_t^h)$;     Test by HANN
14:     $A_c \leftarrow Testing(CANN, m \in S_t^c)$;     Test by CANN
15:     Skip if a channel $\notin S_o$
16:  end
End;
```

the training window slides forward one day at the end of the current day, the logs at t_{a+W+1} become training data while data in day t_a are discarded, that is, and the staring time for training becomes t_{a+1}. Intuitively, if W is too small, there is a lack of training data. Conversely, a larger W might meet higher training cost and noisy old data. It has been demonstrated in [35] that the amount of data has a significant impact on prediction performance in deep learning by LSTM. The appropriate sliding window size W is of importance for the performance of our model. Further discussion will be presented in Sec.VI-B.

C. CLASSIFIED LEARNING BY CHANNEL POPULARITY

In the work, we found the channel popularity has a great impact on the recommending performance. Based on aforementioned **Rule 3** in Sec.III, a channel is classified as hot if its popularity is greater than $\hbar$, otherwise cold. That is, for different viewers, a channel has probably different popularity. We found that a hot channel has better recommendation accuracy than a cold channel with the given neural networks.

To improve the overall recommendation accuracy, we design a classified learning method based on channel popularity, called separated learning (abbr. SL), as described in Algorithm 1. Its core idea involves in following steps. First, to train a hot ANN (called HANN) with hot channels and a cold ANN (called CANN) with cold channels, respectively; and then it will recommend a channel with the trained HANN when the user is watching a hot channel, and do so with the trained CANN for the next recommendation if the user is watching a cold channel. In addition, the PCRS will skip to next if a watching channel is not in the training set, which belongs to the cold boot problem and the discussion is out of the scope of this paper. In theory, the recommendation precision of separated learning is related to the ratio of the hot channels to cold channels and their own precisions in testing set. Here, we give out the accuracy (precision) of using SL method for user u on the dth test day, noted $\mathbb{A}_{\mathbb{SL}}(u, d)$, and $\mathbb{A}_{\mathbb{SL}}$ representing the overall average, as follows,

$$\mathbb{A}_{\mathbb{SL}}(u, d) = \mathbb{A}_c(u, d)\mathbb{P}_c(u, d) + \mathbb{A}_h(u, d)\mathbb{P}_h(u, d) \quad (6)$$

$$\mathbb{A}_{\mathbb{SL}} = \frac{1}{U * D}\sum_{u=1}^{U}\sum_{d=1}^{D}\mathbb{A}_{\mathbb{SL}}(u, d) \quad (7)$$

where D is the total test days, U is the total number of users, $\mathbb{P}_h(u, d)$ denotes the ratio of hot channels, $\mathbb{P}_c(u, d)$ is the ratio of cold channels, and $\mathbb{A}_c(u, d)$ and $\mathbb{A}_h(u, d)$ represent the accuracy of recommending for cold channels and hot channels, respectively. In this work, we found the cold channels actually act as noise and degrade the overall accuracy of recommendation. In practice, high recommendation performance will be achieved after filtering out cold channels. Moreover, the presented separated learning method outperforms the composite learning (called CL). More comparison between CL and SL is demonstrated in Sec.VI-C.

On the contrary, recommending with the knowledge in training set would lead to a Matthew effect [6], [38] of channels access in the future. The presented separate learning method can schedule a trained LSTM model to recommend another cold channel for the latter when he is watching a cold channel. Although the hit may not happen at the next channel switch and there are maybe some lags for the response to the recommended channel, the recall rate increases in obvious, especially for cold channels. On achieving the more effective separate learning recommendation for instant response via some modification of Algorithm1, there is still more work to be done in future.

VI. EXPERIMENTAL RESULT AND ANALYSIS

A series of experiments were conducted to evaluate the impact from a variety of factors related to the proposed recommendation system. Here, we selected 162M records from the original logs of 220k users, after excluding the noise records based on aforementioned **Rule 1** and **Rule 2** in Sec.III. We submitted a subset related to channel switches of 300 IPTV viewers on IEEE Dataport [44]. The proposed PCRS works and rolls day-by-day after the 8th in the month, but the data from the 1st to the 7th are used as an initial training set. In comparison to prior methods, we conduct several groups of experiments to recommend top-K channels.

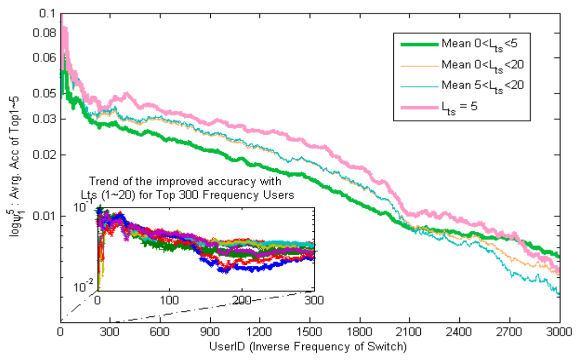


FIGURE 8. **The average improvement of Top1-Top5 A_{cc} with $j = L_{ts}$. Y represents the logarithm of the average improvement of A_{cc} and noted $log\psi_1^5$, x denotes the user ID in the decreasing order of the switching frequency. The pink curve refers to $L_{ts} = 5$, the green curve denotes the logarithm of the mean of $\psi(:, j)$ when j∈ (0, 5), the cyan is that when j∈ (5, 20), and the yellow is when j∈ (0, 20). We zoom in the top 300 users at the bottom-left, where color represents the result from a real L_{ts}.**

We set epoch to be 7, batch size 5, and learning rate of SGD 0.01, respectively, as main parameters of the LSTM neural networks used in the following experiments. And we intend to investigate the impact on recommendation accuracy (actually referring to the precision (A_{cc}) in the paper) from the length of sequence (L_{ts}), the width of training data window (W), learning methods based on the channel popularity, channel ID compression, etc.

A. IMPACTS OF INPUT SEQUENCE LENGTH

The construction of an input sequence is significant and sophisticated in a LSTM model. In order to explore the impact of input sequence length, many experiments have been conducted for learning and testing with real user behavioral logs. In fact, we found the sensitivity to L_{ts} varies with the amount of training data. The more abundant switches, the more remarkable impact from the change of L_{ts}. On the contrary, it has no significant effect on improving the performance if the training data of a user is too sparse. To clarify this fact, we define an evaluation metric, noted as ψ, which is formulated as $\psi(i, j) = \frac{1}{i}\sum_{u=1}^{i} \Delta Acc(u, j)$, here $\Delta Acc(u, j) = \{Acc(u|_{L_{ts}=j}) - Acc(u|_{L_{ts}=1})\}$ means the improvement of recommendation accuracy for user u when increasing L_{ts} from 1 to j. $\psi(i, j)$ represents the average improvement of the first i users. Lower frequency users have less sensitivity for the change of L_{ts}, as shown at the bottom-right portion of the curve in Fig.8. It is thus reasonable for the LSTM to predict the next channel for a viewer using the five channels she/he has watched in the recent past. More demonstration on L_{ts} is shown at the bottom of Fig.8.

These results demonstrated the philosophy on LSTM: 1) the sequence length influences the recommending performance to some extent, but a longer sequence is not always better than the shorter for predicting; 2) the user frequently watching TV is more sensitive to the change of sequence; 3) sparse user data have a poor and non-determinate contribution on the improvement of recommendation performance if nothing is taken into account except for the change of sequence length. It is therefore of necessity to find new chance for unpopular channels in other aspects.

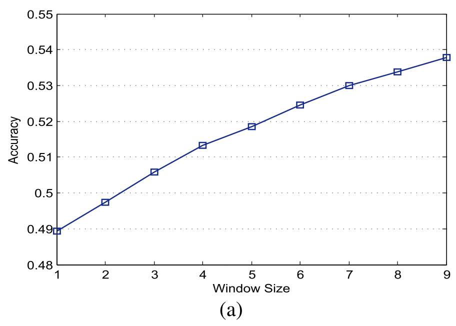
B. IMPACTS OF TRAINING WINDOW WIDTH

The experimental result on the LSTM model shows that the performance increases with the growth of width of training windows, but the increase ratio slows down. The width of the training window is denoted as W in units of days in PCRS. We change W from 1 to 9 for LSTM with $L_{ts} = 5$. As shown in Fig.9 (a), the accuracy increases accordingly with the window size. However, the accuracy increase rate slows down with the increase of W. Meanwhile, the larger the window size, the longer the training time. We illustrate the training time cost and the accuracy together in Fig.9(b). Here, the actual computation time has been already normalized. It's obvious that the system consumes the lowest time and obtains the lowest accuracy when $W = 1$, due to the insufficiency of training data. A larger W means more data for training. The highest accuracy is achieved at window size of $W = 9$, and the relative improvement of accuracy is about 10% over $W = 1$ (i.e., a direct gain of 5%), but at the price of increasing more than 200% training time over $W = 1$. As a compromise scheme, we choose $W = 7$ to balance the accuracy improvement and training time cost. On the whole, the increase of W has a positive impact on the accuracy but leads to much higher computation cost, and the experimental result of the LSTM shows the relative improved ratio is close to 10%. As for the accuracy, Fig.9(a) says the same trend as demonstrated in the work [35].

C. COMPARISON WITH OTHER METHODS

We harnessed LSTM and investigated its applicability in the PCRS. Several traditional channel recommendation methods described in Sec.IV-B are used to compare with the LSTM recommendation. To further compare their recommendation performance, we conduct some experiments using these baseline methods again for popular channels recommendation, namely PS, PP, SVD, and SVD PW (Pairwise). The result is shown in Fig. 10. Because we provide data for the score matrix by the same means of PP and then make matrix decomposition by SVD, the result of the SVD method is similar to that of PP. As for the recommendation accuracy, the LSTM outperforms the others. There are more comparison experiments and analyses on several other recommendation methods and some fusion schemes in our prior work [5].

We empirically found the channel popularity significantly influences the recommendation performance, an ideal design on separate learning is therefore to be presented at Sec.V-C with a classified learning algorithm, i.e. SL Algorithm 1, for every one of users in order to enhance the performance of PCRS. To further study recommendation accuracy for hot and cold channels, we conducted a series of experiments on Combined Learning (CL) and Separated Learning (SL) respectively. In the experiments, we adopted **Rule 3** to distinguish hot and cold channels. A boxplot shows the ratios of



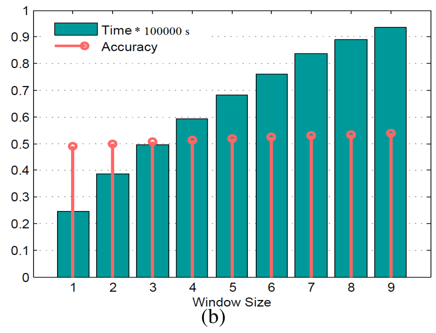


FIGURE 9. The accuracy and time varied with *W* training with the data during 7*24 days. (a) Accuracy. (b) Accuracy vs Time cost.

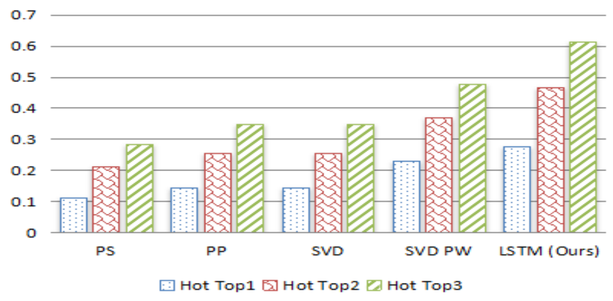


FIGURE 10. The accuracy for hot channels: PS, PP, SVD, SVD PW, and LSTM.

hot channels among the selected 300 users data [44] during 24 testing days in Fig. 11(a). The four types of experiments with original output labels, noted as Hot, Cold, SL and CL, were conducted and the recommendation accuracies are shown in Table 1, in which Hot and Cold represent recommending with a LSTM trained by only hot channels or by only cold channels, respectively. Tons of experiments demonstrate that the separated learning model significantly outperforms the others in precision and recall but an extra price of hit lag. The average accuracy of SL method for user i, noted $A_{SL}(i)$, is calculated by Eqt.(6) and the improvement ratio of SL to CL is defined as $IR_{sl2cl} = (A_{SL} - A_{CL})/A_{CL}$ shown in the fifth column of Table 1. And Fig. 11(b) shows the per-user Top-3 accuracy with LSTM from user1 to user300. The red, blue and green curves represent the accuracy for hot channels with HANN, cold channels with CANN and all channels with the SL method, respectively. It is clear that while the ANN is quite accurate for hot channel recommendation, its accuracy for cold channels is not good enough. Fig. 11(c) compares the average accuracy when recommending all channels using CL versus using SL, from which we can see SL often outperforms CL unless under few situations when the popularity of majority of channels in testing day has occasionally an abrupt change. The gain of average accuracy for Top-3 recommendation using SL over using CL is 19.41%. The result shows SL brings a significant contribution to the gain of accuracy (about 20%).

In Fig. 12, we compare with the accuracies of hot, cold, and all channels using SVD, SVD Pairwise, and LSTM at

TABLE 1. Accuracy: hot, cold, CL, SL, original labels.

Top \ Type	Cold	Hot	CL	SL	IR_{sl2cl}
Top1	9.14%	**25.50%**	15.56%	**17.75%**	14.07%
Top3	24.05%	**58.54%**	35.24%	**42.08%**	19.41%
Top5	34.68%	**80.86%**	48.81%	**58.83%**	20.53%

kind configuration, respectively. Here, we found LSTM significantly outperforms the others when recommending for hot and all channels, but it has no obvious advantage when recommending for cold channels. Reasoning the experimental result, we understand that deep learning by LSTM with a historical switch sequence brings a great contribution to personalized channel recommendation if there are enough logs data for training, while pairwise channels used by SVD actually are a sequence in length of 2, and the recommendation methods with SVD can lead to a better result for cold channels than for hot channel because of using more mutual information among other users. How to improve the performance of cold boot channel and unpopular channels should be further investigated in the future channel recommender systems.

D. IMPACTS OF CHANNEL LABEL SPACE

We found a compact channel label space leads to better recommendation effectiveness than sparse one in LSTM-based live channel recommendation systems. Based on aforementioned channel label compression in Sec.V-A.3, the accuracy with compact labels for training outperforms that with the original labels in our adopted LSTM models, which is shown in Fig.13(a) and Fig.13(b). Fig.14 shows the gain of compact labels over the original labels, which is defined as $IR_C = (A_C - A_O)/A_O$, here A_C is the accuracy with compact labels from Fig. 13(a) while A_O comes from Fig. 13(b). Through the experiments, we found compact labels have a positive impact on the accuracy over the original labels, and the improvement is ranked in descent as [Cold, CL, SL, Hot]. This result demonstrates that compact labels benefit most the learning of cold channels with CANN and the gain is more than 10% over the original labels, and the impact

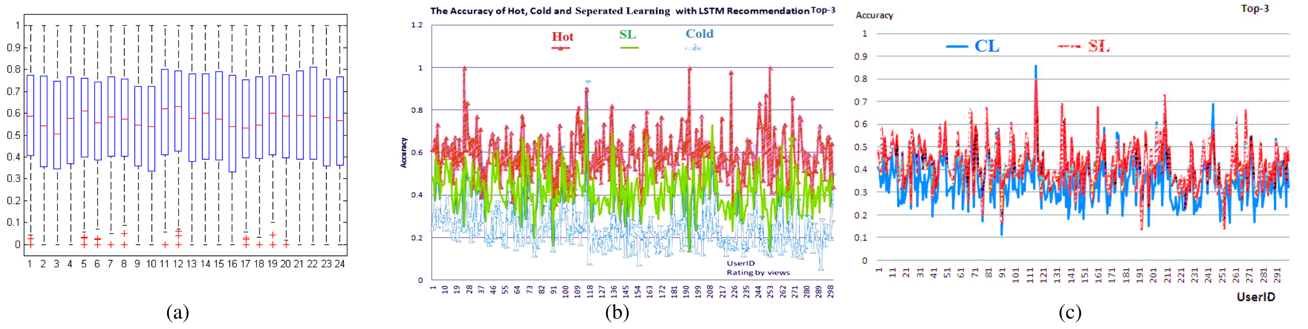


FIGURE 11. **The gain of separated learning (SL) over combined learning (CL). (a) The boxplot of hot channels' percentage. (b) LSTM accuracy for hot, cold and SL. (c) Combined learning vs separated learning.**

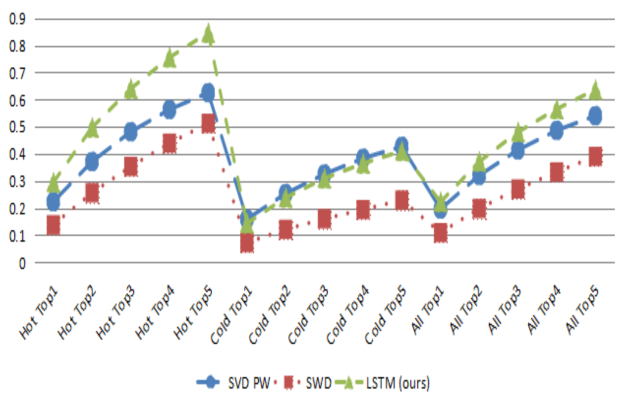


FIGURE 12. **The accuracy for hot, cold, and all channels, with SVD, and SVD PW, and LSTM (ours).**

on HANN is not as significant, the improved ratio is no more than 5%. The improvement from compact labels also decreases with the number of watched channels and the ratio of hot channels within HANN, SL and CL systems. As shown in Fig.14, the more candidates from Top1 to Top5, the less improvement of accuracy, because A_O related to original label space increases if more candidates are recommended. However, cold channels in CANN have much more increase and possibly lead to more gain because the original A_O of cold channels is smaller. These characteristics are pretty different form the aforementioned research. Thus, it can be seen that compact labels benefit sparse representation of cold channels. Numerical results are given in the tables of Fig.13.

Following the analysis in Sec.VI-B that the width of training window has a positive impact on the accuracy, Fig.13(c) and Table 2 further show the accuracy with a large training window size of $W = 24$ days outperforms that with a smaller size of $W = 7$ days. Specifically, data from the first 24 days are used for training, and data from the last 7 days are used for testing, similar to our prior work in [5]. Compared with the result using compact label in the tables of Fig.13(a) with training window size of $W = 7$ days, the recommendation accuracies are improved considerably (about 10%) versus those of $W = 24$, due to the longer training data. This is

TABLE 2. **The accuracy: $L_{ts} = 5$, compact labels.**

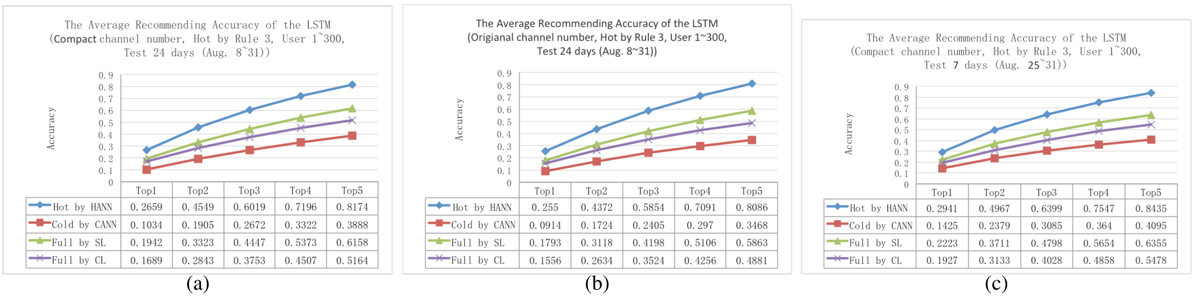
Top \ Type	Cold	Hot	CL	SL	IR_{sl2cl}
Top1	14.25%	**29.41%**	19.27%	**22.23%**	15.34%
IR_C	55.91%	15.33%	23.84%	25.24%	
Top3	30.85%	**63.99%**	40.28%	**47.98%**	19.12%
IR_C	28.27%	9.31%	14.30%	14.02%	
Top5	40.95%	**84.35%**	54.78%	**63.55%**	16.00%
IR_C	18.08%	4.32%	12.23%	8.02%	

accordance with the results in Sec.VI-B. Compared with the result using original labels in Fig. 13(a) at the same context, the recommendation accuracies are also improved about 10% in average versus the results in Table 1, due to the compact labels for sparse representation.

VII. DISCUSSION AND PROSPECTIVE

The main objective of this paper is to demonstrate how a channel switch sequence matches better with LSTM learning and to give unpopular channels a fair chance to be recommended in order to alleviate Matthew effect. All the results exhibit that user behaviors when watching cold channels are more diverse than those when they watch hot channels. The conventional recommendation systems based on previous the probability of consumed items in the training set would lead to critical Matthew effect in future application phase. The presented LSTM model is not only suitable for recommending hot channels but also for recommending the cold ones. The whole trend is shown in Fig.15.

Several issues on the presented method are discussed as follows. First, the proposed framework uses only live channel viewing series which are easier to acquire than the information required in fusion recommendations, and the method based on a channel sequence can be fused into other learning systems. We believe fusion with multiple attributes may further improve the performance. Secondly, the presented PCRS framework is also compatible with other recommendation models, such as CF, Matrix Decomposition, Decision Tree and other ANNs. We will pursue this in our future work. Thirdly, the number of nodes (i.e. sequence length) used for prediction and the width of training windows should be



(a)

	Top1	Top2	Top3	Top4	Top5
Hot by HANN	0.2659	0.4549	0.6019	0.7196	0.8174
Cold by CANN	0.1034	0.1905	0.2672	0.3322	0.3888
Full by SL	0.1942	0.3323	0.4447	0.5373	0.6158
Full by CL	0.1689	0.2843	0.3753	0.4507	0.5164

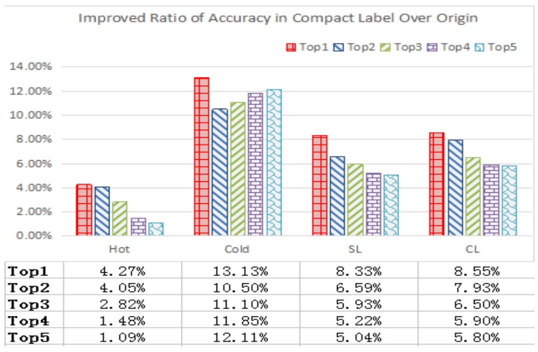
(b)

	Top1	Top2	Top3	Top4	Top5
Hot by HANN	0.255	0.4372	0.5854	0.7091	0.8086
Cold by CANN	0.0914	0.1724	0.2405	0.297	0.3468
Full by SL	0.1793	0.3118	0.4198	0.5106	0.5863
Full by CL	0.1556	0.2634	0.3524	0.4256	0.4881

(c)

	Top1	Top2	Top3	Top4	Top5
Hot by HANN	0.2941	0.4967	0.6399	0.7547	0.8435
Cold by CANN	0.1425	0.2379	0.3085	0.364	0.4095
Full by SL	0.2223	0.3711	0.4798	0.5654	0.6355
Full by CL	0.1927	0.3133	0.4028	0.4858	0.5478

FIGURE 13. Performance comparison between compact labels and original labels. (a) compact label, test the last 24 days. (b) original label, test the last 24 day. (c) compact label, test the last 7 days.



	Hot	Cold	SL	CL
Top1	4.27%	13.13%	8.33%	8.55%
Top2	4.05%	10.50%	6.59%	7.93%
Top3	2.82%	11.10%	5.93%	6.50%
Top4	1.48%	11.85%	5.22%	5.90%
Top5	1.09%	12.11%	5.04%	5.80%

FIGURE 14. The gain of compact label over the origins.

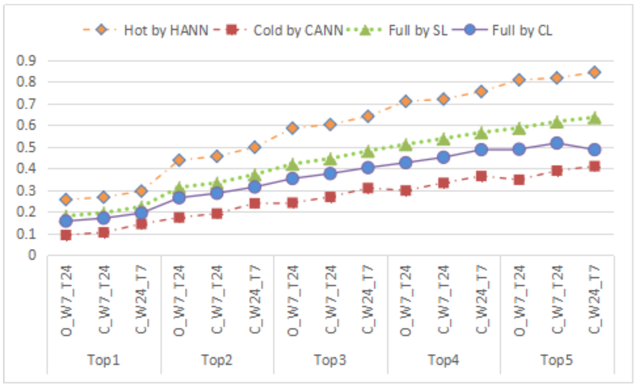


FIGURE 15. The overall performance in multiple scenarios, W7_T24 represents testing in each of the last 24 days with a 7 days sliding window for training while W24_T7 means testing 7 days with 24 days for training, and O and C represent original and compact dataset respectively.

chosen according to the specific applications, and the result from this paper is mined from a real IPTV dataset and can be used as a reference in IPTV channel recommendation systems with the LSTM model. Fourthly, the separate learning for classified recommendation realizes a fair recommendation for channels with different popularity and achieves good precision and recall performance in a variety of experiments, especially for cold channel recommendation. Meanwhile, it incurs an extra price of hit lag because of the randomness in user channel access, namely a user may switch channels multiple times before hitting the recommended one. Furthermore, it may be inferred that multi-classification by popularity probably brings higher precision with k-means or similar clustering algorithms, but more lags and choices may interfere with user viewing experience.

In practice, the SVD-based CF methods must use switch logs of many users and are more suitable to run at server side. For our proposed LSTM methods, for the training phase, to achieve high accuracy, it only needs individual switching logs of a user. Although the computation complexity of LSTM is usually higher than SVD, the computation complexity for learning with the log of one user is generally much less than that of a group of users. On the other hand, the trained LSTM network only needs the target user's latest channel switch sequence to generate online recommendation, so the actual recommendation can be run on the client side. It can therefore be applicable for a mobile device or STB to run both the offline learning and online recommendation. In short, the proposed method is more practical than SVD or other CF methods and can work on the client side without exchanging information with other users.

As for future challenges, real user experience for a channel recommender system is of significance and difficulty to acquire for performance evaluation. We will do our best to design a feedback system for collecting user behavioral data after a recommendation algorithm is deployed in IPTV systems in future. In addition, we will focus on creating more chance for recommending a channel after its cold boot and before being a hot one. We will also pay our attention to the research on the lag of recommendation and cold boot in our future work.

VIII. CONCLUSIONS

In this paper, we proposed a framework of personalized channel recommendation with dynamic data provisioning via deep learning from historical channel switching sequences in IPTV systems. In the proposed framework, a sliding window is used to control the provisioning of training data and a training matrix is constructed with channel switching sequences for LSTM neural networks. The experiments showed that LSTM is suitable for live channel recommendation. It's found that the amount and organization of training data have important impacts on recommendation performance of LSTM. Our study identified several significant factors for the design of IPTV channel recommendation systems, including the compact label space, wide enough training window, adequate predicting sequence length and diversity in channel popularity. Furthermore, we presented a classified recommendation method driven by separate learning based on the popularity of channels in order to alleviate the Matthew effect suffered by the conventional recommendation methods. We showed that hot and cold channels have quite different characteristics and one should use separated LSTMs, instead of a combined LSTM, to generate fair recommendations for channels. For future work, we will plan to make a fusion of LSTM with other proposed channel RSs, and further focus on the balance among precision, recall, response lag, and other metrics of recommendation systems.

ACKNOWLEDGMENT

The authors thank the editors and reviewers for their kind suggestions and review comments, and also thank for the help of Mr. Wu, Songtao, Dr. Zhou, Yipeng and Prof. Wu, Di. The partial data used in the paper were published on IEEE Dataport, 2018. [Online]. Available: http://dx.doi.org/10.21227/H2396N.

REFERENCES

[1] C. Yang and Y. Liu, "On achieving short channel switching delay and playback lag in IP-based TV systems," *IEEE Trans. Multimedia*, vol. 17, no. 7, pp. 1096–1106, Jul. 2015.

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[3] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Comput.*, vol. 12, no. 10, pp. 2451–2471, 2000.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[5] Y. Chenguang, D. Hao, C. Houwei, L. Yong, and Y. Can, "Follow me: Personalized IPTV channel switching guide," in *Proc. 8th ACM Multimedia Syst. Conf.*, 2017, pp. 147–157.

[6] R. K. Merton, "Matthew effect in science," *Science*, vol. 159, no. 3810, pp. 56–63, 1968.

[7] D. H. Park, H. K. Kim, L. Y. Choi, and J. K. Kim, "A literature review and classification of recommender systems research," *Expert Syst. Appl.*, vol. 39, no. 11, pp. 10059–10072, 2012.

[8] O. Phelan, K. McCarthy, and B. Smyth, "Using twitter to recommend real-time topical news," in *Proc. 3rd ACM Conf. Recommender Syst.*, Oct. 2009, pp. 385–388.

[9] N.-H. Liu, S.-J. Hsieh, and C.-F. Tsai, "An intelligent music playlist generator based on the time parameter with artificial neural networks," *Expert Syst. Appl.*, vol. 37, no. 4, pp. 2815–2825, 2010.

[10] L. Canini, S. Benini, and R. Leonardi, "Affective recommendation of movies based on selected connotative features," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 4, pp. 636–647, Apr. 2013.

[11] C. Min, Z. Yin, Q. Meikang, G. Nadra, and H. Yixue, "SPHA: Smart personal health advisor based on deep analytics," *IEEE Commun. Mag.*, vol. 56, no. 3, pp. 164–169, Mar. 2018.

[12] C. Min, H. Yixue, H. Kai, W. Lu, and W. Lin, "Disease prediction by machine learning over big data from healthcare communities," *IEEE Access*, vol. 5, pp. 8869–8879, 2017.

[13] Z. Zhang, H. Lin, K. Liu, D. Wu, G. Zhang, and J. Lu, "A hybrid fuzzy-based personalized recommender system for telecom products/services," *Inf. Sci.*, vol. 235, pp. 117–129, Jun. 2013.

[14] R. Crespo *et al.*, "Recommendation system based on user interaction data applied to intelligent electronic books," *Comput. Hum. Behav.*, vol. 27, no. 4, pp. 1445–1449, 2011.

[15] 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.

[16] C. E. Briguez, M. C. D. Budán, C. A. D. Deagustini, A. G. Maguitman, M. Capobianco, and G. R. Simari, "Argument-based mixed recommenders and their application to movie suggestion," *Expert Syst. Appl.*, vol. 41, no. 14, pp. 6467–6482, 2014.

[17] X. Yang, H. Steck, Y. Guo, and Y. Liu, "On top-k recommendation using social networks," in *Proc. 6th ACM Conf. Recommender Syst.*, 2012, pp. 67–74.

[18] G. Ling, M. R. Lyu, and I. King, "Ratings meet reviews, a combined approach to recommend," in *Proc. 8th ACM Conf. Recommender Syst.*, 2014, pp. 105–112.

[19] G. Ziyu *et al.*, "Document recommendation in social tagging services," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 391–400.

[20] Y. Zhang, "GroRec: A group-centric intelligent recommender system integrating social, mobile and big data technologies," *IEEE Trans. Services Comput.*, vol. 9, no. 5, pp. 786–795, Sep./Oct. 2016.

[21] T. N. Thanh, C. P. Thanh, T. N. Thanh, H. N. Q. Viet, and S. Bela, "Diversifying group recommendation," *IEEE Access*, vol. 6, pp. 17776–17786, Mar. 2018.

[22] 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, 2015.

[23] E. Kim, S. Pyo, E. Park, and M. Kim, "An automatic recommendation scheme of TV program contents for (IP)TV personalization," *IEEE Trans. Broadcast.*, vol. 57, no. 3, pp. 674–684, Sep. 2011.

[24] A. B. Barragáns-Martínez, E. Costa-Montenegro, J. C. Burguillo, M. Rey-López, F. A. Mikic-Fonte, and A. Peleteiro, "A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition," *Inf. Sci.*, vol. 180, no. 22, pp. 4290–4311, 2010.

[25] M. Krstić and M. Bjelica, "Personalized TV program guide based on neural network," in *Proc. 11th Symp. Neural Netw. Appl. Elect. Eng.*, Sep. 2012, pp. 227–230.

[26] S. Pyo, E. Kim, and M. Kim, "Automatic and personalized recommendation of TV program contents using sequential pattern mining for smart TV user interaction," *Multimedia Syst.*, vol. 19, no. 6, pp. 527–542, 2013.

[27] S. Pyo, E. Kim, and M. Kim, "LDA-based unified topic modeling for similar TV user grouping and TV program recommendation," *IEEE Trans. Cybern.*, vol. 45, no. 8, pp. 1476–1490, Aug. 2015.

[28] 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.*, Dec. 2013, pp. 188–193.

[29] 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*, 2016, pp. 183–193.

[30] 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, Oct. 2014.

[31] 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.

[32] Y. Zhu, J. Diao, B. Kang, and S. Liu, "User experience optimization research on personalized push approach of IPTV service based on label of contents," *Telecommun. Sci.*, vol. 30, no. 7, pp. 113–120, Jul. 2014.

[33] S. Martin, N. Hermann, and S. Ralf, "From feedforward to recurrent LSTM neural networks for language modeling," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 23, no. 3, pp. 517–529, Mar. 2015.

[34] Z. Qiu, T. Yao, and T. Mei, "Learning deep spatio-temporal dependence for semantic video segmentation," *IEEE Trans. Multimedia* vol. 20, no. 4, pp. 939–949, Apr. 2018.

[35] T. Ergen and S. S. Kozat, "Efficient Online learning algorithms based on LSTM neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 8, pp. 3772–3783, Aug. 2018, doi: 10.1109/TNNLS.2017.2741598.

[36] D. Tim, L. Benedikt, and Z. Jürgen, "Sequential user-based recurrent neural network recommendations," in *Proc. 11th ACM Conf. Recommender Syst.*, vol. 27, Aug. 2017, pp. 152–160.

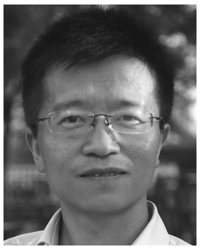
[37] Z. Yu *et al.*, "What to do next: Modeling user behaviors by time-LSTM," in *Proc. 26th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2017, pp. 3602–3608.

[38] M. J. Salganik, P. S. Dodds, and D. J. Watts, "Experimental study of inequality and unpredictability in an artificial cultural market," *Science*, vol. 311, no. 5762, pp. 854–856, Feb. 2006.

[39] M. Cha, R. Pablo, J. Crowcroft, M. Sue, and A. Xavier, "Watching television over an IP network," in *Proc. 8th Usenix/ACM SIGCOMM Internet Meas. Conf.*, Vouliagmeni, Greece, 2008, pp. 71–84.

[40] F. M. V. Ramos, "Mitigating IPTV zapping delay," *IEEE Commun. Mag.*, v. 51, n. 8, pp. 128–133, Aug. 2013.

[41] C. Ye, C. Zhao, Y. Yang, and C. Fermüller, "Lightnet: A versatile, standalone MATLAB-based environment for deep learning," in *Proc. ACM Multimedia Conf.*, 2016, pp. 1156–1159.
[42] M. Bjelica, "Experiment with user modeling for communication service retrieval," *IEEE Commun. Lett.*, vol. 12, no. 10, pp. 797–799, Oct. 2008.
[43] D. M. Harris and S. L. Harris, *Digital Design and Computer Architecture*, 2nd ed. San Francisco, CA, USA: Morgan Kaufmann, 1988, p. 129.
[44] S. Ren and C. Yang, "Channels switch sequences of 300 IPTV viewers in a month," *IEEE Dataport*, 2018, doi: 10.21227/H2396N.
[45] R. Cañamares and P. Castells, "Should i follow the crowd?: A probabilistic analysis of the effectiveness of popularity in recommender systems," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2018, pp. 415–424.



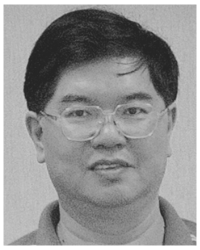
CAN YANG (M'12) received the B.D., M.D., and Ph.D. degrees from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 1994, 1997, and 2002, respectively. He held a post-doctoral position at the College of Computer Science and Technology, HUST, from 2002 to 2004. He was a Visiting Scholar at the Department of ECE, New York University, in 2013. He has been an Associate Professor and a Professor-in-Advance with the College of Computer Science and Engineering, South China University of Technology, since 2005 and 2013, respectively. His current research interests include cross-media computing, mobile video systems, and online recommendation systems. He is a Senior Member of CCF. He received the Science and Technology Progressive Award of Guangdong Province in 2011 and 2015, and the Science and Technology Innovation Award of SARFT of China in 2012, respectively.

SIXUAN REN received the bachelor's degree from the College of Computer Science and Technology, Hainan University, Haikou, China, in 2015, and the masterŠs degree from the College of Computer Science and Engineering, South China University of Technology, in 2018. She is good at LSTM recommendation. Her current interest focuses on multimedia recommendation systems with deep learning.

YONG LIU (F'17) received the bachelor's and master's degrees in automatic control from the University of Science and Technology of China, in 1994 and 1997, respectively, and the Ph.D. degree from the Electrical and Computer Engineering Department, University of Massachusetts, Amherst, in 2002. He joined the New York University Tandon School of Engineering as an Assistant Professor in 2005. He is currently an Associate Professor with the Electrical and Computer Engineering Department, New York University. His general research interests lie in the modeling, design, and analysis of communication networks. His current research involves P2P systems, overlay networks, network measurement, online social networks, and recommender systems. He was a recipient of the IEEE Communications Society Best Paper Award in Multimedia Communications in 2008, the IEEE Conference on Computer and Communications Best Paper Award in 2009, and the ACM/USENIX Internet Measurement Conference Best Paper Award in 2012.

HOUWEI CAO received the Ph.D. degree in electronic engineering from The Chinese University of Hong Kong in 2011. She is an Assistant Professor with the Department of Computer Science, University of New York Institute of Technology. She was an Adjunct Professor with the Computer Science and Engineering Department, New York University Tandon School of Engineering. She was a Post-Doctoral Fellow at the University of Pennsylvania from 2011 to 2014 and Tufts University from 2014 to 2015. She was also an Insight Data Science Fellow in 2015. She is currently an Assistant Professor with the Department of Computer Science, University of New York Institute of Technology. Her main areas of research are signal processing, machine learning, and data mining and their applications in human-centric data analysis, with emphasis on developing computational methods for speech recognition, text analysis, affect detection, and healthcare analysis. She is a member of the International Speech Communication Association and the Association for the Advancement of Affective Computing. She was a recipient of the Audio-Visual Emotion Recognition Challenge in 2012. She has served as a Reviewer for numerous conferences and journals, including Interspeech, ISCSLP, O-COCOSDA, the IEEE TRANSACTIONS ON SPEECH AND AUDIO PROCESSING, the IEEE TRANSACTIONS ON AFFECTIVE COMPUTING, the *IET Transaction on Computer Vision*, and *Speech Communications*.

QIHU YUAN received the bachelor's degree from the Department of Computer Science and Technology, Lanzhou Jiaotong University, Lanzhou, China, in 2015. He is currently pursuing the master's degree with the College of Computer Science and Engineering, South China University of Technology. He is good at matrix factorization and tensor analysis. His current interest focuses on multimedia recommendation systems, video communication applications, and deep learning.



GUOQIANG HAN received the B.Sc. degree from Zhejiang University in 1982, and the master's and Ph.D. degrees from Sun Yat-sen University in 1985 and 1988, respectively. From 1997 to 1999, he held a post-doctoral position at The University of Tokyo, Japan. He was the Dean of the College of Computer Science and Engineering, South China University of Technology (SCUT), Guangzhou, China, from 2007 to 2017. He is currently a Professor with the College of Computer Science and Engineering, SCUT. He is also the President of the Cantonese Computer Association. He has undertaken many projects, such as national support plan, the National Natural Science Funds, the Natural Science Foundation of Guangdong Province, Guangdong Province Science and Technology Research, and a batch of enterprise cooperation projects. He has published over 100 research papers. His research interests include multimedia, computational intelligence, machine learning, and computer graphics. He was a recipient of the Science and Technology Awards of Guangdong Province trice and the Science and Technology Innovation Award of SARFT of China in 2012, respectively.