IPTV Channel Switch Recommendation with Attention Mechanism

Guangyu Li, Lina Qiu, Chenguang Yu, Houwei Cao, Yong Liu, *Fellow, IEEE*, and Can Yang

***Abstract*—Internet Protocol TV (IPTV) normally has the advantage of providing far more TV channels than the traditional TV services, while as the other side of the coin it has the problem of information overload. Users of IPTV usually have difficulties finding channels matching their interests. In this paper, facilitated with a large IPTV dataset, we analyze the dynamics of users' channel switching behaviors and discover various patterns that can contribute to building a more accurate channel switch recommendation system. Based on user behavior analysis, we develop several base and fusion recommender systems that generate in real-time a short list of channels for users to consider whenever they want to switch channels. A deep neural network model that consists of a "Recommender System Attention (RS Attention)" module and a "Channel Attention" module capturing the static and dynamic user switching behaviors is also developed to further improve the recommendation accuracy. Evaluation on the IPTV trace demonstrates that our fusion recommender can achieve 41% hit ratio with only three candidate channels and our attention neural network model further pushes it up to 45%.**

***Index Terms*—IPTV, Recommender System, Realtime Recommendation, Fusion Method, Attention Mechanism, Neural Networks.**

I. Introduction

Smart TVs and highly developed Internet video streaming services have helped Internet Protocol TV (IPTV) gain more popularity among users than the tradition TV service. However, the long existing problem of *"which channel to watch"* not only remains unsolved but even becomes worse since there are more online contents available for users to choose. Compared to the Over-the-Top (OTT) content providers, such as Netflix or Hulu, who are equipped with powerful Recommendation Systems (RSs), this problem has become an obvious weak point for the IPTV service providers. Recommendation Systems have been through a rapid development process in both industry and academia. RSs are now widely used by companies such as Amazon, Netflix, and Spotify to enhance customer experience and improve revenue. Advanced machine learning techniques, from the well-known collaborative filtering to more recent deep neural networks, have been developed to efficiently generate large-scale recommendations to millions of customers. With the user channel watching history data and abundant computational power, IPTV providers have all the technical means to develop customized RSs to generate recommendations to users on-the-fly. But not much RS adoption has been observed so far. Electronic Program Guide (EPG), which provides a long list of channels embedded in a hierarchical menu, is still the common practice for IPTV providers. Personalized recommendation is never the aim of general EPG. A general EPG can only provide a long channel list that contains all channels' descriptions and/or sample videos for users to choose from. The burden of decision is now on the shoulders of the users, which is annoying since the list usually contains hundreds of channels. A channel RS on the other hand can be designed to provide a short list of channels tailored to a target user's personal taste. It is also more desirable to generate realtime channel recommendation on-the-fly: instead of recommending a long list of channels when a user turns on TV and using it for the whole watching session, a new short channel list is generated and popped up on TV whenever the user initiates channel switching from her remote.

IPTV contents are different from online movies and on-demand videos because they are mainly organized in program sequences whose broadcasting schedules are mostly fixed and periodic, e.g. daily news and TV series. Some programs are only broadcasted once without fixed schedules, e.g. live sports events and coverage of emerging events. One way of recommending channels is to weaken the impact of "channels" and recommend a channel x if we predict the user will be interested in the program currently being broadcasted at channel x. However, a few obstacles need to be overcome before we can use this method. First, for most RSs (especially those using Collaborative Filtering (CF)) we rely on the fact that users consume online products asynchronously, which makes it possible to predict user-item ratings by leveraging previous user ratings and user-user similarities. Different from online movie or video-on-demand providers, the "cold item" problem for IPTV is much more severe since in IPTV system, all users consume the items at the same time and most of the items can be treated as "cold items" without previous user ratings. To solve this issue, one would have to maintain and analyze various metadata, e.g. the description of old programs watched by users. One also needs to access detailed metadata for new programs to be broadcasted. However, detailed program metadata are not always available, especially for one-time and unplanned programs. Secondly, the traditional RSs that only target on program-based channel recommendation cannot determine the timing for switching, i.e., for a user who is in the middle of watching a program when to switch to another channel. This is a critical issue for realtime channel recommendation.

G. Li, L. Qiu, C. Yu and Y. Liu are with the Department of Electrical and Computer Engineering, New York University, Brooklyn, NY, 11201 e-mail: guangyu.li@nyu.edu.

H. Cao is with the Department of Computer Science, New York Institute of Technology, New York, 10023 e-mail: hcao02@nyit.edu.

Y. Can is with the Department of Computer Science, South China University of Technology, Guangzhou, China, 510006 e-mail: cscyang@scut.edu.cn

In this paper, we develop channel RSs that generate realtime channel recommendations to guide user channel switching in IPTV systems. Our RSs only need access to channel watching sequences of users, and don't need any program metadata, nor involve any program content analysis. Specifically several base RSs that can make recommendations individually such as global and personal channel popularity, personal schedule, channel transition pattern, etc., are proposed. Base RSs are then combined through a fusion process at the last stage to provide recommendations with improved accuracy. Different fusion mechanisms are explored, such as ranking-based approaches and data partitioning. Finally, a deep neural network based RS model is developed to further improve the performance. The model consists of two sub modules: the "RS Attention module" and the "Channel Attention module". The former one captures the static user switching behaviors reflected by the channels that users stayed in for long time, the latter one captures the dynamic user switching behaviors including the random tuning and channel exploration when the users deviate from their routine channel watching. Through evaluation on a real IPTV user channel switching trace, we demonstrate that it is possible to generate accurate realtime channel recommendations by only mining user channel watching sequences. The best fusion RS achieves an impressive hit ratio of 41% when only three channels are recommended. The performance can be further pushed to around 45% with the proposed attention mechanisms.

The rest of the paper is organized as follows. In Section II, we discuss the related work on RSs for TV channels. In Section III, we characterize IPTV user channel switching behaviors and formalize the realtime channel recommendation problem and its workflow. In Section IV, we develop our base and fusion RSs. Attention-based neural network model is presented in Section V. In Section VI, we evaluate the performance of the proposed models. Section VII concludes the paper.

II. Related Work

Before IPTV gains its popularity and data gathering becomes easier through Internet, the behaviors of TV users are insufficiently studied in the literature. The emergence of OTT content not only stimulates the studies of OTT user behaviors in several areas but also enables the widely adoption of recommender systems among OTT providers, such as Spotify, Netflix and YouTube. Both content popularity and user behaviors have been studied for OTT video-on-demand services, e.g, [1], [2] and [3]. Measurement and modeling of video watching time in a large-scale Internet video-on-demand system was presented in [1]. In [2], user behaviors for live and on-demand content were compared for an IPTV system delivering both types of content. Treating TV channels as OTT content, Cha et al. [4] was able for the first time to characterize a series of channel viewing properties, such as viewing sessions, channel popularity, user geographical distribution, and channel switching behaviors for a large IPTV network. Later, Qiu et al. [5] also conducted IPTV channel popularity analysis and focused on its temporal dynamics. Different from the previous studies, where users' channel switching patterns are mostly overlooked, we dig deep into the user channel switching behaviors for the latent information about users' channel/program/schedule preferences.

User experience is highly related to the prediction accuracy of the next channel the user may switch to. Studies in [6], [7] and [8] used channel popularity based content pre-fetching to reduce channel switching delay, which is much longer in IPTV than in the traditional TV. Meanwhile, other studies used channel switching prediction to simply improve user experience of finding interesting channels to watch. Various RS algorithms, e.g., [9], [10], [11], have been proposed to match users' personal interests with the huge amount of content choices. In the TV domain, most RSs were built to address the program recommendation problem [12]. After the first EPG was introduced by Das et al. [13], a series of rule-based, statistical or machine learning approaches have been proposed for TV program RS, e.g., [14], [15], [16], [17], [18], [19], [20]. Yang et al.[20] described a TV program RS with collaborative filtering approach based on statistic and affinity propagation. Chang et al. [21] proposed a TV channel RS based on the feedback loser tree (FLT) algorithm. In [19], Turrin et al. combined three simple popularity metrics for channel recommendation and demonstrated the significance of considering temporal context of channels. Different from the above studies we aim at recommending channels to users and weaken the concept of "program" because of the minimum data overhead and the advantages of exploiting the dynamic behaviors of users.

Attention mechanism based neural networks are developing very fast recently due to its remarkable effectiveness in the several domains, including Natural Language Processing (NLP) and Image Recognition, e.g., [22], [23], [24]. Several studies used attention based neural networks to mimic the allocation of human preferences. In [25], an attention based neural network is proposed to mimic users' preferences on different sub models at different time for recommending articles. Some of our earlier results were published in a conference paper [26], that solely uses fusion method to combine all sources of information to generate final recommendation. In this paper, on top of [26], we further develop a novel attention mechanism based neural network model which mimics users' potential attention allocation process to replace the fusion method. Attention based neural network can learn more flexible "fusion functions" with less limitation.

III. IPTV Channel Recommender System

A. Channel Switching and Terminologies

IPTV can be mainly classified into three groups based on the type of services [27]: live television, time-shifted (replay enabled) television, and Video on Demand (VoD). In this paper, we consider an IPTV system that provides all three types of services at the same time. We focus on an IPTV system where clients can access live shows, replay a TV show that was broadcast hours or days ago, and browse and view contents in a stored media catalogue. Our study is based on a dataset provided by a major IPTV service provider for the

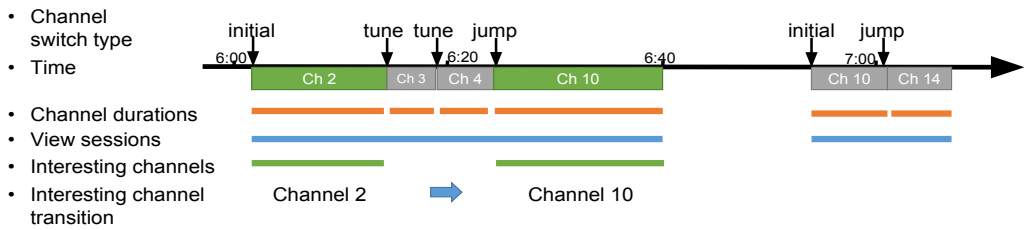


Fig. 1: A sample of raw channel-watching log and the derived user behavior statistics.

metropolitan area of Guangzhou, China. It consists of user channel watching logs for the entire month of August 2014. Each log is a five-tuple:

$$\{userid, channelid, duration, title, description\}$$

There are totally 222K users, 172 channels, and 73M logs.

Figure 1 illustrates sample channel-watching logs of a user. From the logs, we can obtain the user's channel switching sequence, and calculate her watching duration for each channel. We can further derive the following user behavior data.

- **Watching Session** is defined as a period during which the user turns on her TV, watches a sequence of channels, till she turns off the TV. In theory, there shouldn't be any gap for two sequential channel watching activities in a session. But in some cases, the user may turn her TV off and on in just a few seconds. We still treat it as a consecutive watching session. In our trace, to handle the quick "on-off-on" activities, we cluster all channel watching logs with time gap less than 10 seconds into one watching session. Figure 1 consists of two sessions, during each of which the user watched four and two channels respectively.
- **Channel Switching Type**: A user can reach a channel through three types of channel switching: she starts with the first channel appeared after she turns on the TV (*initial*); she intentionally jumps from her current channel to another target channel by typing the channel number on her remote (*jump*); she randomly navigates to the next or previous channel by pressing the channel up or down button on her remote (*tune*). Our trace does not have user remote action logs. Instead, we classify a channel switching into *jump* or *tune* by simply checking whether the id of the channel switched to is adjacent to the id of the previous channel. Channel switches in Figure 1 are labeled correspondingly.
- **Interesting Channels**: The time that a user spends on a channel reflects her interest in the channel. We define an interesting channel as a channel being watched continuously by the user for a duration longer than some threshold T. Two interesting channels are marked with green color in Figure 1, when we choose $T = 10$ minutes.
- **Transition between Interesting Channels**: It is important for us to understand a user's interest transitions, which is defined as the transition between two adjacent interesting channels. It is not necessarily an observed channel switching sequence. In Figure 1, the transition from channel 2 to channel 10 never shows up as a channel switching sequence, but it might suggest that the user tends to watch channel 10 after watching channel 2, even though she watched channel 3 and 4 briefly in between.

Since the raw data has inconsistent entries and out-of-date information, we pre-process the data by eliminating vague entries and inactive users. The cleaned data consist of 135K users, 172 channels and 51M logs.[1]

B. Recommendation Task and Workflow

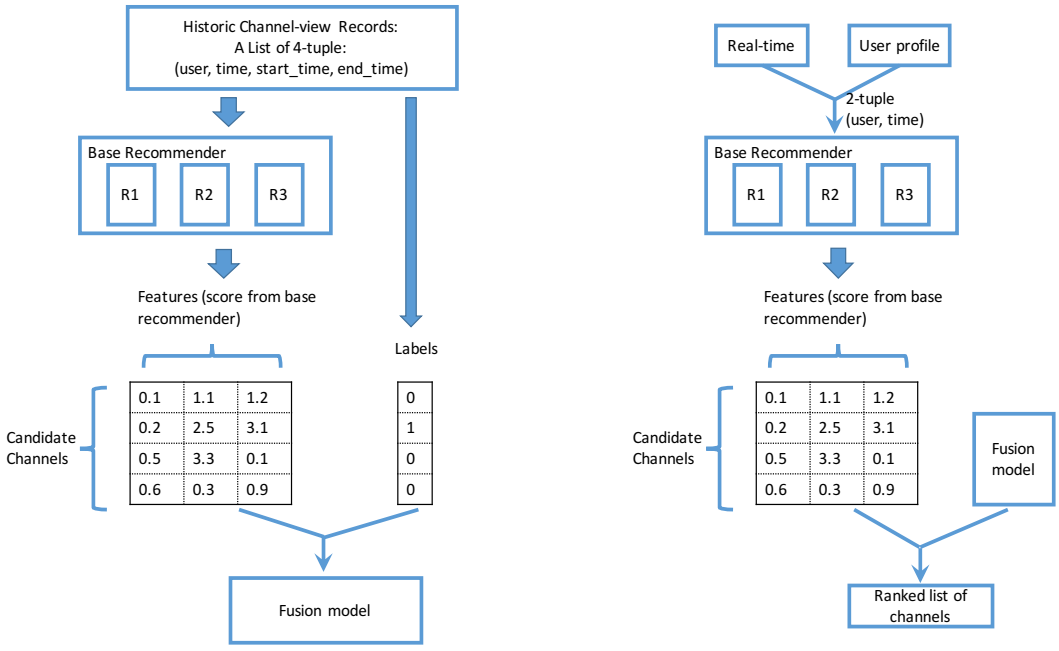
Although channel RSs can generate realtime recommendation at any time, we choose the user-initiated channel-switching actions to be the recommendation trigger. For the example in Figure 1, recommendation should be generated at all channel switching moments after the user finished watching *channel 2* and before she finds *channel 10*. We will also compare recommendation performance for different types of channel switchings. Given historic channel-watching logs of a large number of users, RS will generate a score $score_{c,u,t}$ for each candidate channel c at time t, and return a list of k channels $\widehat{\mathbf{C}}(u,t)$ with the highest scores. The top-k channel list gets a hit if it includes the channel c that is indeed watched by user u for at least $T = 10$ minutes immediately after t.

There are mainly two workflows for our proposed system as in Figure 2: *training flow* and *recommendation flow*. During the training phase, base RSs are prepared/trained based on user channel-watching logs. Each base RS will generate a score for each (candidate channel, user) pair. These scores will serve as features to train fusion RS models as will be described in Section IV-C. During the recommendation phase, we use (user ID, time) as the only input, and feed it to the trained base RSs to generate features, which are fed to the trained fusion RS models to generate realtime channel recommendations.

IV. Base and Fusion Recommenders

In this section, we first develop six *base recommenders* based on simple statistics of user channel watching history. We then investigate different *fusion recommenders* that combine scores generated by base RSs using different fusion methods to produce the final recommendation list.

[1]The data cleaning for this paper is slightly different from the conference version [26], the results presented later in this paper will be slightly different from their counterparts in [26] even with the same setting.



(a) Training Phase

(b) Recommendation Phase

Fig. 2: Work Flows for the Proposed Realtime Channel Recommender System

A. *Base Recommenders*

The logs contain various types of information that can be used for channel recommendation. For each user, we mostly focus on the channels that she watched for a time period more than a threshold (e.g. 10 minutes). We develop six base RSs, each of which focuses on one type of information. From logs of user channel-watching, we can easily derive various user-channel relation features, such as $\mathcal{U}(c,t)$: the set of users watching channel c at time t; $d(u,c,\mathcal{T})$: the total time that user u spent watching channel c within some time period $\mathcal{T}$, which can be either contiguous (e.g., the previous week) or non-contiguous (e.g., all 9pm - 10 pm time slots of the previous week).

1) Current Global Popularity. This method recommends the most popular channels among all users at any given moment t, i.e., the channels watched by the most users. The score of each channel c for user u at time t is defined as: $score^{gp}_{c,u,t} = |\mathcal{U}(c,t)|$, where $|\cdot|$ denotes the set size. This score is common for users.

2) Historical Personal Popularity. This method recommends the channels watched the most by the target user during a history window (e.g. last week). The score of channel c for user u at time t is defined as: $score^{pp}_{c,u,t} = d\left(u,c,[t-\Delta,t)\right)$, where Δ is the history window size (e.g., a week).

3) Personal Schedule. This method recommends channels based on a user channel watching history at specific time slots within a history window. The score of each channel c for user u at time t is defined as:

$$score^{ps}_{c,u,t} = d\left(u,c,[t-\Delta,t)\cap\mathcal{S}(t)\right),$$

where $\mathcal{S}(t)$ is the time slots that t belongs to. For example, if the history window is one month, we use hourly slots to define schedule, then $t = 8:14pm$ belongs to the $[8pm, 9pm)$ hourly slot. The channels watched the most by a user between $8pm$ and $9pm$ in the past month will be recommended to the user.

4) User-based Collaborative Filtering. This method recommends channels that are being watched by the most similar users, a.k.a. the nearest neighbors. Given a set $\mathbf{C}$ of available channels, for each user u, we define her channel-watching duration vector during a period $\mathcal{T}$ as $\mathbf{D}^{\mathcal{T}}_u$, with $\mathbf{D}^{\mathcal{T}}_u[c] = d(u,c,\mathcal{T})$, $\forall c \in \mathbf{C}$. The similarity $sim(u,v,t)$ between user u and v at time t is the cosine similarity of their channel-watching duration vector $\mathbf{D}^{[t-\Delta,t)}_u$ and $\mathbf{D}^{[t-\Delta,t)}_v$ during the window before t:

$$sim(u,v,t) = \frac{\mathbf{D}^{[t-\Delta,t)}_u \cdot \mathbf{D}^{[t-\Delta,t)}_v}{\left\|\mathbf{D}^{[t-\Delta,t)}_u\right\| \left\|\mathbf{D}^{[t-\Delta,t)}_v\right\|}$$

We can then find the k-nearest-neighbors of user u as $\mathcal{U}^{knn}_u$ and calculate the user CF recommendation score as:

$$score^{ucf}_{c,u,t} = \sum_{v\in\mathcal{U}^{knn}_u} d(v,c,[t-\Delta,t)). \tag{1}$$

5) Personal Channel Transition. This method recommends channels based on each user's channel transition pattern. From history viewing logs, we can derive the channel transition probability: given the previous interesting channel c' watched by user u, the transition probability that the next interesting channel watched by u is c can be calculated as:

$$p_u(c|c') = \frac{|\mathbf{S}_u(c'\to c)|}{|\mathbf{S}_u(c'\to *)|}$$

where $\mathbf{S}_u(c'\to c)$ is the set of user u's interesting channel switching actions from c' to c, and $\mathbf{S}_u(c'\to *)$ is the set of user u's interesting channel switching actions from c' to

any other channel. The recommendation score is defined as: $score^{ct}_{c,u,t} = p_u(c|l(u,t))$, where $l(u,t)$ is the last interesting channel watched by u prior to t.

6) Tune Probability. Different from the previous five base RSs, this method focus on user channel tuning behaviors. It takes into account all channels that a user visited, regardless of how long she stayed with them. Specifically, it recommends channels based on the last channel ID the user *spanned over* and the probabilities that the user tunes to the adjacent channels. Note that this recommender tries to predict the likelihood of a user reaches a channel but not necessarily stays (e.g. for over 10 minutes) at the channel. Given a user u and the previous spanned channel s, let s^o be the next spanned channel which resides a certain offset o away from channel s. The probability that the next channel spanned by u to be s^o given the previous channel is s can be calculated as:

$$p_u(o) = \frac{\sum_s \mathbf{N}_u(s \rightarrow s^o))}{\sum_s \mathbf{N}_u(s \rightarrow *)},$$

where $\mathbf{N}_u(s \rightarrow s^o)$ is defined as the number of times user u is observed in the channel watching logs to switch from any channel s to another channel with offset of o from s, o takes values of -5,-4,-3,...3,4,5 in practice. $\mathbf{N}_u(s \rightarrow *)$ is the number of times user u switch from any channel s to another channel with any offset. At time t, given the last channel user u watched before t is $c(t)$, and the offset between the last channel c and next channel c is o, the recommendation score is then defined as:

$$score^{tp}_{c,u,t} = p_u(o), o = -5, ..., 5$$

B. Base RS Performance

Since all base RSs need enough history data to obtain stable statistics, we test the performance of base RSs by letting them generate recommendations for all channel switches in the last seven days of our trace. For a channel switch initiated by user u at time t, base RS x calculates scores $score^x_{c,u,t}$ for all target channels based on user channel viewing history up to time t as described above. All channels will be ranked based on their scores, and the top k channels with the highest scores will be returned as the top-k recommendation list generated by base RS x. If the next channel user u watched for more than 10 minutes after t is in the recommendation list, we call a recommendation hit, otherwise it is a miss. The top-k hit ratio is averaged over all switches from all users in the test set. The top-k hit ratios for all base RSs are reported in Figure 3. The "*personal popularity*" and the "*personal schedule*" base RSs are generally the best among all base RSs. The "*channel transition*" RS performs better when the recommendation list is short ($k < 2$) and the "*personal popularity*" RS performs the best when the recommendation list is longer ($k > 4$).

C. Fusion Recommenders

Since each base RS only captures limited information about user channel watching behaviors, more accurate recommendation can be generated by efficiently utilizing all information captured by all RSs. Now we study fusion RSs that combines

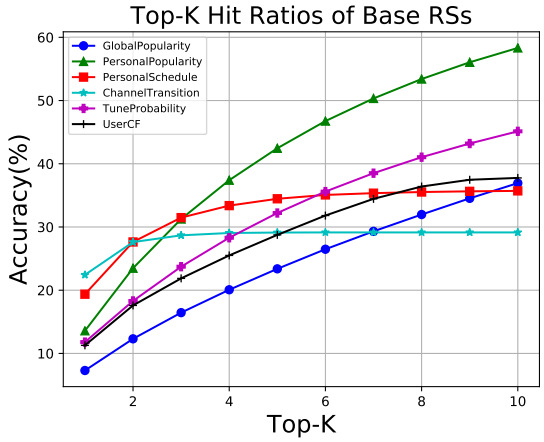


Fig. 3: Top-k Hit Ratio of Base Channel Recommender Systems.

scores generated by base RSs using some fusion function $\mathcal{F}(\cdot)$ to produce the final recommendation list.

$$score_{c,u,t} = \mathcal{F}(\langle score^{gp}_{c,u,t}, score^{pp}_{c,u,t}, score^{ps}_{c,u,t}, score^{ucf}_{c,u,t}, score^{ct}_{c,u,t}, score^{tp}_{c,u,t} \rangle)$$

The performance of fusion RS is largely determined by the design of $\mathcal{F}(\cdot)$, including fusion models, data partition methods, and ranking approaches.

Prediction Models. The goal of a fusion RS is to predict whether a channel will interest a user based on the scores obtained by base RSs. This is a typical binary classification problem, and a lot of existing prediction models can be adopted. We mainly explored three of them: *Logistic Regression (LR)* [28], *Support Vector Machine (SVM)* [29] and *Random Forest (RF)* [30]. For each model, in the training phase, whenever a user switches channel, we obtain the candidate channels from all base RSs, and use their associated scores as feature values. We then assign a binary label to each candidate channel, depending on whether the user actually watched the channel after the switch. Using training data, we obtain binary classification models (LR, SVM, RF), which will be used to generate fusion scores for channels, and consequently recommendation lists in the test phase.

Ranking Approaches. We also investigated two approaches of ranking channels: *pointwise* and *pairwise*. The pointwise approach ranks all channels directly based on their fusion scores. The pairwise approach, on the other hand, predicts if a channel is more interesting than another channel based on their feature values. To conduct pairwise ranking, original samples are transformed into sample pairs. For example, at certain time, there are three original samples $(s_{ch1}, true)$, $(s_{ch2}, false)$, $(s_{ch3}, false)$, where s_{ch1} is the score vector of channel 1 and $true/false$ is the label of whether the user watched the channel. Then we can transform these three samples into two sample pairs: $(s_{ch1} - s_{ch2}, true)$, $(s_{ch3} - s_{ch1}, false)$, where the $true$ label represents channel 1 is more interesting than channel 2, and the $false$ label represents channel 3 is less interesting than channel 1. Pairs between samples with original negative labels are ignored. Then we train binary

classification models to predict the relative ranking between channel pairs based on the difference between their score vectors. To generate recommendation list in the test phase, we first estimate the relative ranking among all candidate channel pairs. If channel A has ranks higher than channel B, it will get one vote. Finally, the channels with most votes will be placed at the top of the recommendation list.

Data Partition. We also need to consider whether and how our data should be grouped for model training. For examples, on one hand, if we train a model for all users over all time, the model granularity may be too coarse; on the other hand, if we partition data according to users/hour and train per-user/per-hour models, our training data will become too sparse for training. We studied three ways of data partition: *no partition*, *per-user partition* (different models for different users), and *per-hour partition* (different models for different hour-of-day).

The performance of fusion RSs will be reported in Section VI-A.

V. Attention-based Channel Recommenders

In this section, we show how the attention mechanism can be used to improve channel recommendation accuracy.

A. Dynamic-attention-based Approach

Instead of directly estimating the probability of a user stays in a channel, we want to first narrow down the searching range of the interesting channels, and then keep refining this range by exploring more information until we can generate a short recommendation list. Similar approaches were proposed in other research areas. For instance, in object detection in natural images, the "Region of Interest (ROI)" is pre-calculated as a rough estimate of the coordinates of the bounding box, and later this bounding box is refined to improve the detection accuracy [31]. In Neural Machine Translation (NMT), the recent dominant methods adopted an"attention mechanism based encoder-decoder framework" to achieve the superior performance [24]. The gain is mainly due to the ability of assigning different weights to previous samples in time series or sequential data. In our IPTV system, we can also model two types of attention. The first one is the "RS attention". It is assumed that users have several principles about how to reach their interested channels. Although different users have different principles, we further assume that they share some common principles, such as "what is the most popular channel right now?", or "is it time to watch my favorite show?" which can be easily captured by our base RSs previously introduced (e.g. the "Global Popularity RS" and the "Personal Schedule RS"). We can then use the weighted combination of the base RS scores to mimic the decision process of users' channel switching. The "RS attenton" is the importance weights users pay upon the six base recommenders implicitly. The second attention is the "channel attention". Just as the machine translation and the auto-reply email system, our goal is to predict the next word (or value) in a sentence (or time series). Because of the high similarity between our problem and the problem defined in [24], we propose to capture the second type of attention using a model similar to [24], but with customized model structure for IPTV system.

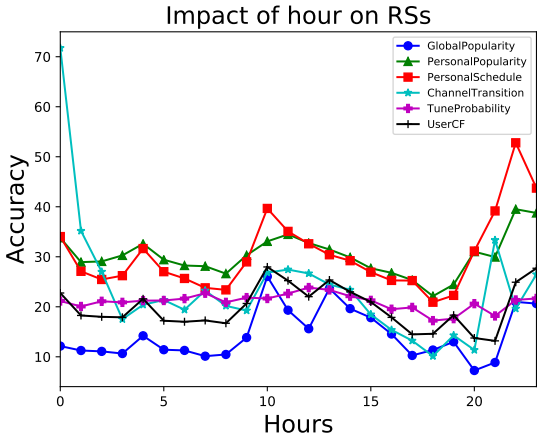


Fig. 4: Performance of base RSs with different hour of day.

One key question is how to capture the two types of user attention in different context. The context can be external environment information, such as the hour of day when a channel switching occurs, or user behavior data, such as users' recent channel switching sequence. So we propose a neural network model that contains two sub modules to capture the two attention we described previously. The model's specialty lies in its awareness of the context and its capability of dynamically predicting users' attention within the context. In V-B we introduce the module that handles the type of attention users pay to different base recommenders within a context defined by two manually extracted features, and in V-C we show the sub module which captures the attention users pay on channel sequence by using a customized seq2seq model.

B. Attention-based RS Fusion

To learn the dynamic attention allocation we need to feed features that can contribute to a user's decision. In practice, we find two features that can affect the attention a user pays on different base RSs (which correlates to her channel switching principles): the hour-of-day and the number of switches spanning over non-interesting channels since the last interesting channel. The hour-of-day impacts the decision process because users may change their searching strategies in response to the ongoing schedules of the TV programs. For example, as show in Fig. 4, for the most hours of the day, the "Global Popularity RS" has the lowest performance among RSs, however at 10:00am, its performance increases rapidly and surpasses the performance of the "Tune Probability RS". At other hours of the day, the relative accuracy order and the differences among the six RSs are also changing. Another feature is the number of switches spanning over non-interesting channels since the last interesting channel (we will call it as "the number of switches" for the rest of this paper). If a user cannot find her interested channels after many switches, the laziness may drive her back to her comfort zone, i.e. what she used to watch before. Fig. 5 shows the effect of the number of switches on base RSs. We can see that the relative accuracy order

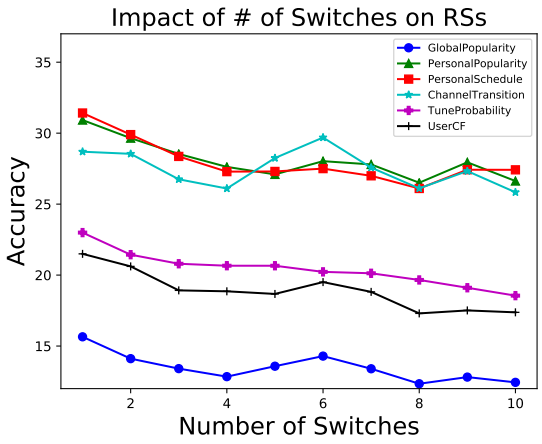


Fig. 5: Performance of base RSs with different number of switches.

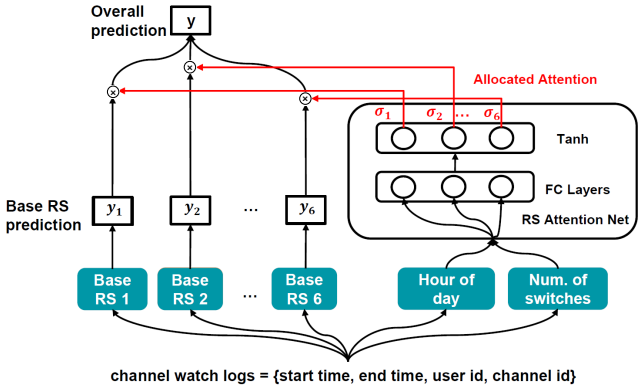


Fig. 6: The structure of RS Attention Network.

of "personal popularity", "personal schedule" and "channel transition" keeps changing with different numbers of switches.

In practice, we represent the hour-of-day as a 24-d one-hot vector and quantify the number of switches using 3 ranges: $1 \sim 3, 4 \sim 10, 10+$. A 3-d one-hot vector is then generated and appended to the 24-d one-hot vector of the hour-of-day. The input of our algorithm is then a 27-d vector $\vec{o}$ containing the above two sources of information. We then design a "RS Attention Network" to capture both the results of six base RSs and the effect of the hour-of-day and the number of switches. The structure of "RS Attention Network" is illustrated in Fig. 6. The input $\vec{o}$ goes through L fully connected layers and the "RS Attention Network" can be formulated in a hyperbolic tangent form as below:

$$\vec{\lambda_0} = \vec{o} \cdot \vec{w_0} + \vec{b_0} \tag{2}$$

$$\vec{\lambda_i} = \overrightarrow{\lambda_{i-1}} \cdot \vec{w_i} + \vec{b_i}, i = 1, 2, ..., L-1 \tag{3}$$

$$\vec{\sigma} = Tanh(\overrightarrow{\lambda_{L-1}}) = \frac{e^{2 \cdot \overrightarrow{\lambda_{L-1}}} - 1}{e^{2 \cdot \overrightarrow{\lambda_{L-1}}} + 1} \tag{4}$$

$$\vec{y} = \sum_{i=1}^{6} (\vec{y_i} \cdot \vec{\sigma i}) \tag{5}$$

where $\lambda_i, i = 0, 1, ..., L-1$ is the output of the i-th fully

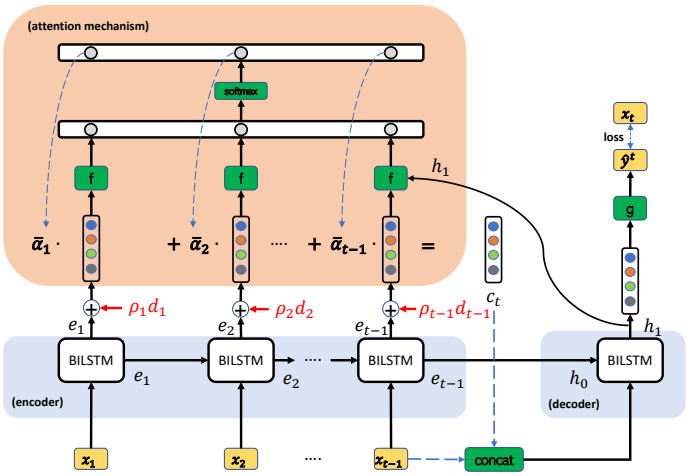


Fig. 7: Encoder-decoder based Channel Attention Model with Special Attention Mechanism Customized for IPTV Switch Prediction Problem. The sequence length is fixed to 10 at each time t.

connected (FC) layer among L FC layers in total, $\vec{\sigma}$ is the allocated attention on six base RSs and $\vec{y_i}, \vec{y}$ are the normalized output of six base RSs and the overall score for recommendation respectively. The output of each intermediate layer (from the input layer to the last hidden layer) in this module is normalized by a batch normalization layer to prevent covariance shift. $\vec{w_i}$ and $\vec{b_i}$ are tunable parameters in FC layers. The use of "Tanh" function makes sure that the contribution of any base RS can be positive or negative.

C. Attention-based Sequence Model

Since we can only exploit the previous channel number sequence to make predictions, a natural question would be how users set the weights on the previous channels spanned over, and how they combine the weighted experience of previous channels to find the next channel to switch to. For example, some switches may only be the intermediate states in the course of finding the next interesting channel, thus should be assigned with less weights. Note that this problem is very similar to the machine translation problem [32] where the goal is to translate a sentence in one language to one in another language. The difference is that in our problem all generated "translations" have a fixed size of one word (the next channel number). The dominant models (e.g. "Seq2Seq model" [33]) for machine translation are based on recurrent or convolutional neural networks in an encoder and decoder configuration to generate the translation. The best performing models also connect the encoder and decoder through an "attention mechanism" [34]. Due to the common nature of our problem and the machine translation problem, we propose to use a similar encoder-decoder model with special attention mechanism customized for our channel switch problem. This model serves as a module to the final model for prediction of the interesting channels. Figure 7 shows the details of the module which is explained as the following:

- *Encoder.* We replace the vanilla LSTM module in the traditional Seq2Seq model with Bidirectional LSTM module (BiLSTM) to exploit the reversed sequence information. Let the input channel number sequence be $\{ch_{t-1}, ch_{t-2}, ...ch_{t-10}\}$. Each channel number ch_i is mapped to a one-hot vector $\vec{x_i} \in R^C$, where C is the number of unique channel numbers in system. Our goal is to output the next channel to switch to ($\hat{y}^t$). The BiLSTM module is then run $t-1$ times and the hidden states $\{e_{t-1}, e_{t-2}, ...e_{t-10}\}$ are stored.
- *Decoder.* The decoder contains a BiLSTM module that will run one time step (sequence length fixed to 1) since we only predict the next channel number. The initial hidden state of the BiLSTM equals to the last hidden state of the encoder e_{t-1}. The input of the BiLSTM module (besides the initial hidden state h_0) is a concatenated vector containing 1) the most recent one-hot channel number $\vec{x_{t-1}}$, and 2) the context vector calculated from the attention mechanism which will be explained later. We apply a function g which is implemented with several fully-connected hidden layers to the output of the BiLSTM. The output of g is a vector of the same size of input $\vec{x_i}$. We then apply a softmax function to squeeze the output of g into a vector of probabilities and a argmax function to find the most probable channel for the next switch.

$$\vec{h_1} = BiLSTM(\vec{h_0}, [\vec{x_{t-1}}, \vec{c_t}]) \quad (6)$$

$$\vec{s_1} = g((\vec{h_1})) \quad (7)$$

$$\vec{p_1} = softmax(\vec{s_1}) \quad (8)$$

$$\vec{i_1} = argmax(\vec{p_1}) \quad (9)$$

- *Attention Mechanism.* The attention mechanism is a way to force the decoder to connect to all hidden state in encoder instead of only learning from the last hidden state of the encoder as in vanilla Seq2Seq model. The context vector c_t used in the decoder is calculated as follows:

$$\vec{\alpha_t} = f(h_1, e_{t'} + \rho_{t'} \cdot d_{t'}) \in R \quad (10)$$

$$\vec{\bar{\alpha}} = softmax(\vec{\alpha}) \quad (11)$$

$$\vec{c_t} = \sum_{t'=1}^{t-1} \bar{\alpha_{t'}} \cdot (e_{t'} + \rho_{t'} \cdot d_{t'}) \quad (12)$$

$$t' = t-1, t-2, ...t-10 \quad (13)$$

For the choice of f function, there are several widely used versions such as [24] and [35]. In our model, we just use the "dot" function for the simplicity of computation, i.e.

$$f(h_1, e_{t'}) = \vec{h_1} \cdot \vec{e_{t'}} \quad (14)$$

Unlike the traditional attention mechanism in machine translation problem, we refine the attention weights $\{\alpha_1, \alpha_2...\}$ with the information of how long a user stayed in previous channels. The idea is that the duration that a user spent on previous channel $\{d_1, d_2...\}$ also reflect her strength of attention on each previous channel. We then combine the duration with the hidden states using a weight vector so that the input hidden vector to attention module becomes $\{e_{t-1} + \rho_{t-1}d_{t-1}, e_{t-2} + \rho_{t-2}d_{t-2}...\}$.

D. Unified Attention-based Recommender

We create a unified model for end-to-end training and testing by combining the outputs of both RS attention network $\vec{y}^{RS}$ and channel attention network $\vec{y}^{ch}$:

$$\vec{y}^{final} = softmax(\beta_0 * \vec{y}^{RS} + \beta_1 * \vec{y}^{ch} + b * \vec{\mathbf{1}}), \quad (15)$$

where β_0, β_1 and b are tunable parameters, and $\vec{y}^{final}$ is the output of softmax function representing the probability distribution for the user to stay in all channels.

Finally we use $\vec{y}^{final}$ as the final likelihood vector that contains the probability of any channel to be the next "interesting channel". With this final likelihood we can rank the channels and choose the K channels with the highest probability as our top-K recommendation. The parameters of the model used in practice are shown in Table I.

Parameters	Settings
RS Attention Network	fc layer w/ 64 unites × 3
BiLSTM	128 units × 3
sequence length of BiLSTM	10
batch sizes	8
function g	fc layer w/ 64 unites × 3
Learning rate	10^{-4}
Optimizer	Adam

TABLE I: Parameters in the model

VI. Evaluation of Model Fusion and Attention

A. Fusion RS Performance

For the fusion model, to search for the best fusion setting, we enumerate all combinations of the previous three fusion design dimensions. For each setting, we test and train a fusion RS using a subset of our dataset which contains $13,284$ users. Again, data from the first 24 days are used for training, the last 7 days for testing. In Table II, we compare the recommendation accuracy in terms of hit ratio. We also include two baseline methods: 1) "Best Single" is the best base RS when recommending top 1/3/5 channels; 2) "Score Sum" uses the sum of scores of all base RSs as the final score for a channel. Some settings such as SVM model and per-user partition are skipped in Table II due to their bad performance. Among all the other settings, we can see random forest (RF) models generally outperform other models, and the setting of per-hour pairwise random forest model is the best among them all. It achieves significant improvement compared with the baseline methods.

To evaluate whether RSs can provide complementary information, leave-one-out experiments are performed. We pick one base RS at a time and compare the performance of fusion RS without it. As in Figure 8, all base RSs can generally improve the final performance (except some top-1 cases) and the "personal popularity" and "tune probability" methods are the most important two. One observation is that removing the "personal schedule" and "channel transition" do not have too

TABLE II: Fusion Model Comparison.

	Best Single	Score Sum	LR	RF	LR per-hour	RF per-hour	LR Pair	RF Pair	RF Pair per-hour
Top 1	16.8%	16.2%	18.4%	19.4%	18.4%	18.9%	18.1%	19.1%	**19.9%**
Top 3	31.9%	34.8%	37.1%	38.9%	37.2%	39.5%	37.1%	39.5%	**41.6%**
Top 5	44.3%	46.7%	48.3%	50.6%	48.5%	51.1%	48.6%	50.8%	**51.5%**

	Best of Fusion (%)	Channel Attention (%)	RS Attention (%)	Channel&RS Attention (%)	Gain over Fusion (%)
Top 1	19.9	15.7	19.4	**20.2**	1.5
Top 3	41.6	32.6	40.6	**44.3**	6.5
Top 5	51.5	42.7	52.3	**57.5**	11.7

TABLE III: Performance Comparison of Fusion and Attention Networks with Different Settings.

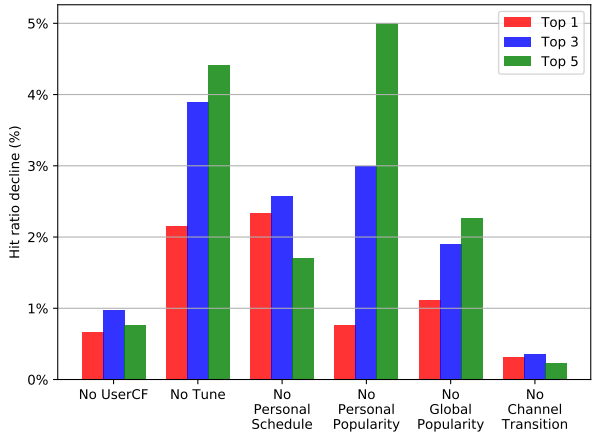


Fig. 8: Performance of Fusion RS after Removing One Base RS.

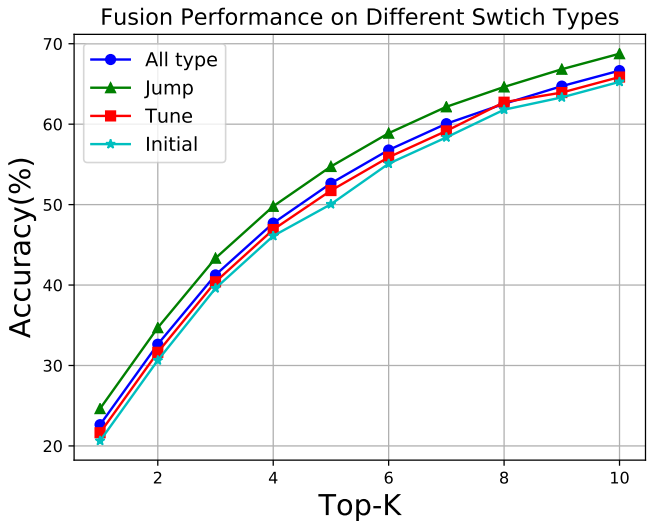


Fig. 9: Top-k Hit Ratio of Fusion Channel RS for Different Switching Types.

much negative effect on fusion RS as indicated in Fig. 8, even though "personal schedule" and "channel transition" are the second and third most effective base RS when operating alone, as shown in Fig 3. This suggests that the complementary information these two base RSs provide to the fusion RS is not much in general. The potential performance decline caused by removing them could be compensated by other base RSs to some extent (not completely). On the other hand, we notice that although the individual performance of tune probability RS seems mediocre in Fig. 3, the complementary information it provides to the fusion RS is significant. This is because the "tune probability" can capture the exploration process which cannot be captured by other RSs. We also found that the fusion RS generally perform better for "jump" type of channel switching as illustrated in Figure 9. It is not surprising because users usually have some ideas about which channel may interest them when they directly "jump" to a channel. Therefore "jump" switching is more predictable than "tune". *This result suggests that our RS can catch users' realtime personal preferences. Meanwhile, our RS is still accurate for "tune" switchings. Indeed, the utility of RS might be even higher for "tune" switchings, since this is when a user is less clear about what she should watch, and needs guidance to find interesting channels.*

B. Attention Networks' Performance

For the attention networks, in Table III, we show the recommendation accuracy (hit ratio) with RS attention, Channel attention, and combined RS-Channel attention. We compare them with the previous fusion methods. For example, the column corresponding to "Channel Attention" means only using previous channel switch sequence with duration spent on each channel as features to recommend, "Channel&RS Attention" means using the unified model to generate recommendations. Table III shows that by applying attention networks the performance of top-1, top-3 and top-5 can be improved by up to 1.8%, 10.1%, 11.7% respectively comparing to the best performance of fusion method (RF pairwise per-hour model) while they use similar set of features. The performance of using "RS Attention" alone to control the weights among six base RSs can match the performance of the best fusion method and it also provides an interface that enables the model to cooperate with the "channel attention module" in an end-to-end training framework which can achieve better performance. Besides the difference of techniques, the major difference between dynamic attention based approaches and

the static behavior based approaches is that the former ones use the "non-interesting" channel logs as additional information to make recommendations. Because in our dataset, a large proportion of the users have stable switching behaviors which can be best reflected by top-1 hit ratio. The dynamic RS approaches (attention networks) do not outperform much (by only 1.8%) over the static approach (fusion method). However, when the users have no clear choices of the next channel, they tend to explore the channels. This is the situation where dynamic attention based approaches perform better (outperform the fusion methods by up to 12%). We also tried to combine fusion method with the "channel attention module", the accuracy cannt match the unified neural network model. The reason may be the end-to-end training can make the "RS attention" and the "channel attention" module to cooperate. Due to space limit, We don't show the detailed results for this experiment.

We also tried different configurations of models by varying combinations of the following choices: 1) BiLSTM or vanilla LSTM, 2) with or without Attention Mechanism and 3) if Attention is applied, with or without duration information. Fig. 10 shows the comparison of the results in different configurations. It can be shown that with BiLSTM the performance is

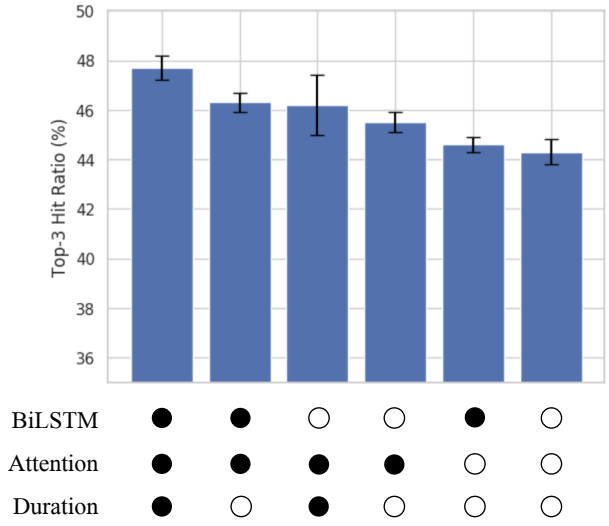


Fig. 10: Different Experiment Configurations with Combinations of Components.

slightly improved, while adding Attention Mechanism to the model brings larger improvement. The duration information as "the hard attention" to the model can improve the performance by up to 5% (relative). The full model with all 3 choices enabled can outperform the vanilla model (encoder-decoder with only LSTM) by up to 8%.

VII. Conclusions

In this paper, we studied channel switching recommendation for IPTV users. Using a large IPTV user channel watching trace, we first developed six base RSs that generate channel recommendations using basic user-channel features, such as user schedule, personal hot channels, channel transition patterns, as well as classic recommendation methods, such as global popularity and user-based CF. We improve the accuracy of base RSs using different fusion methods. A deep neural network model that consists of a "Recommender System Attention" module and a "Channel Attention" module capturing the static and dynamic user switching behaviors is also developed to further improve the recommendation accuracy Through evaluation, we demonstrated that our fusion RS and attention models outperform individual base RSs and can accurately guide user channel switching by using extremely short recommendation lists. Our proposed RSs are suitable for real-time channel recommendation in practical IPTV systems.

References

[1] H. Abrahamsson and M. Nordmark, "Program popularity and viewer behaviour in a large tv-on-demand system," in *Proceedings of the 2012 ACM conference on Internet measurement conference*. ACM, 2012, pp. 199–210.

[2] Y. Chen, B. Zhang, Y. Liu, and W. Zhu, "Measurement and modeling of video watching time in a large-scale internet video-on-demand system," *Multimedia, IEEE Transactions on*, vol. 15, no. 8, pp. 2087–2098, 2013.

[3] N. Liu, H. Cui, S.-H. G. Chan, Z. Chen, and Y. Zhuang, "Dissecting user behaviors for a simultaneous live and vod iptv system," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 10, no. 3, p. 23, 2014.

[4] M. Cha, P. Rodriguez, J. Crowcroft, S. Moon, and X. Amatriain, "Watching television over an ip network," in *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*. ACM, 2008, pp. 71–84.

[5] T. Qiu, Z. Ge, S. Lee, J. Wang, Q. Zhao, and J. Xu, "Modeling channel popularity dynamics in a large iptv system," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 1. ACM, 2009, pp. 275–286.

[6] E. Lee, J. Ku, and H. Bahn, "An efficient hot channel identification scheme for iptv channel navigation," *Consumer Electronics, IEEE Transactions on*, vol. 60, no. 1, pp. 124–129, 2014.

[7] C. Yang and Y. Liu, "On achieving short channel switching delay and playback lag in ip-based tv systems," *Multimedia, IEEE Transactions on*, vol. 17, no. 7, pp. 1096–1106, 2015.

[8] S. Zare and A. G. Rahbar, "Program-driven approach to reduce latency during surfing periods in iptv networks," *Multimedia Tools and Applications*, pp. 1–13, 2015.

[9] R. H. Keshavan, A. Montanari, and S. Oh, "Matrix completion from noisy entries," *Journal of Machine Learning Research*, vol. 11, no. Jul, pp. 2057–2078, 2010.

[10] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.

[11] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," in *Proceedings of KDD cup and workshop*, vol. 2007, 2007, pp. 5–8.

[12] D. Véras, T. Prota, A. Bispo, R. Prudêncio, and C. Ferraz, "A literature review of recommender systems in the television domain," *Expert Systems with Applications*, vol. 42, no. 22, pp. 9046–9076, 2015.

[13] D. Das and H. ter Horst, "Recommender systems for tv," in *Recommender Systems, Papers from the 1998 Workshop, Technical Report WS-98-08*, 1998, pp. 35–36.

[14] L. Ardissono, C. Gena, P. Torasso, F. Bellifemine, A. Difino, and B. Negro, "User modeling and recommendation techniques for personalized electronic program guides," in *Personalized Digital Television*. Springer, 2004, pp. 3–26.

[15] M. Chen and C. Yang, "Private recommendation system based on user social preference model and online-video ontology in interactive digital tv," in *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2012 4th International Conference on*, vol. 2. IEEE, 2012, pp. 260–263.

[16] S. H. Hsu, M.-H. Wen, H.-C. Lin, C.-C. Lee, and C.-H. Lee, "Aimed-a personalized tv recommendation system," in *European Conference on Interactive Television*. Springer, 2007, pp. 166–174.

[17] W.-P. Lee and J.-H. Wang, "A user-centered control system for personalized multimedia channel selection," in *Consumer Electronics, 2004 IEEE International Symposium on*. IEEE, 2004, pp. 430–435.

[18] B. Smyth and P. Cotter, "Surfing the digital wave," in *International Conference on Case-Based Reasoning*. Springer, 1999, pp. 561–571.

[19] R. Turrin, A. Condorelli, P. Cremonesi, and R. Pagano, "Time-based tv programs prediction," in *1st Workshop on Recommender Systems for Television and Online Video at ACM RecSys*, 2014.

[20] Y. Yang, C. Liu, C. Li, Y. Hu, Y. Niu, and L. Li, "The recommendation systems for smart tv," in *Computing, communication and networking technologies (ICCCNT), 2014 international conference on*. IEEE, 2014, pp. 1–6.

[21] H.-Y. Chang, C.-C. Lai, and Y.-W. Lin, "A fast svc-based channel-recommendation system for an iptv on a cloud and p2p hybrid platform," *The Computer Journal*, vol. 57, no. 12, pp. 1776–1789, 2014.

[22] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, 2015, pp. 2048–2057.

[23] W. Yin, H. Schütze, B. Xiang, and B. Zhou, "Abcnn: Attention-based convolutional neural network for modeling sentence pairs," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 259–272, 2016.

[24] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[25] X. Wang, L. Yu, K. Ren, G. Tao, W. Zhang, Y. Yu, and J. Wang, "Dynamic attention deep model for article recommendation by learning human editors' demonstration," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17. New York, NY, USA: ACM, 2017, pp. 2051–2059. [Online]. Available: http://doi.acm.org/10.1145/3097983.3098096

[26] C. Yu, H. Ding, H. Cao, Y. Liu, and C. Yang, "Follow me: Personalized iptv channel switching guide," in *Proceedings of the 8th ACM on Multimedia Systems Conference*. ACM, 2017, pp. 147–157.

[27] A. Punchihewa and A. M. De Silva, "Tutorial on iptv and its latest developments," in *2010 5th International Conference on Information and Automation for Sustainability (ICIAFs)*, 2010, pp. 45–50.

[28] G. A. Seber and A. J. Lee, *Linear regression analysis*. John Wiley & Sons, 2012, vol. 329.

[29] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[30] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[31] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[32] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002, pp. 311–318.

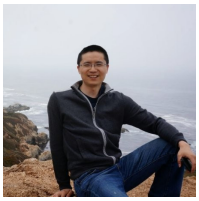
[33] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 5998–6008. [Online]. Available: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[35] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

Guangyu Li is a Ph.D. candidate at the Electrical and Computer Engineering department of New York University. He received his bachelor and master degree in the field of automatic control from Electrical and Computer Engineering department at Tianjin University, Tianjin, China in July 2011 and 2014 respectively. His general research interests lie in data mining methods in networking and recommender system.

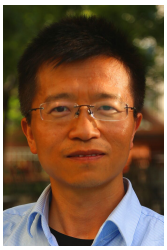
Lina Qiu is a Ph.D. student at the Department of Computer Science at Boston University. She received her master degree in Computer Science from New York University in 2019, and her bachelor degree in Computer Science and Electronics from University of Edinburgh in 2017. Her general research interests lie in data systems and information retrieval.



Chenguang Yu received a BS and MS degree in Biology from University of Science and Technology of China in 2006 and 2013, and a PhD degree in Electrical Engineering from New York University in 2016. He is a research scientist in Facebook. His research spans the area of machine learning and data mining with an emphasis on online social network based recommender systems. His current work is focused on data science related research for Instagram.

Houwei Cao is an assistant professor in the Department of Computer Science at New York Institute of Technology (NYIT). She was an adjunct professor at the Computer Science and Engineering Department of the Tandon School of Engineering of New York University before joining NYIT. She obtained her Ph.D. degree in Electronic Engineering from the Chinese University of Hong Kong in 2011, and worked on an RGC grant on automatic speech recognition of Cantonese-English code-mixing speech. She was a postdoctoral fellow at University of Pennsylvania from 2011 to 2014 and worked on an NIH grant on the computational quantification of emotion in faces and voice for neuropsychiatry. While at Tufts University from 2014 to 2015, Cao worked on an NSF grant on social robotics and Parkinson's disease. She was also an Insight Data Science fellow in 2015.

Yong Liu is an associate professor at the Electrical and Computer Engineering department of New York University. He joined NYU-Poly as an assistant professor in March, 2005. He received his Ph.D. degree from Electrical and Computer Engineering department at the University of Massachusetts, Amherst, in May 2002. He received his master and bachelor degree in the field of automatic control from the University of Science and Technology of China, in July 1997 and 1994 respectively. His general research interests lie in modeling, design and analysis of communication networks. His current research include P2P systems, overlay networks, network measurement, online social networks and recommender systems. He is the winner of ACM/USENIX Internet Measurement Conference (IMC) Best Paper Award in 2012, IEEE Conference on Computer and Communications (INFOCOM) Best Paper Award in 2009, and IEEE Communications Society Best Paper Award in Multimedia Communications in 2008. He is a Fellow of IEEE.



Can Yang is an associate professor since 2005 and a professor in advance since 2013 at the School of Computer Science and Engineering of South China University of Technology. He was a postdoctor in the School of Computer Science and Technology from August 2002 to October 2004 at Huazhong University of Science and Technology (HUST) Wuhan, China. He received his Ph.D. and bachelor from the Department of Electronic Science and Technology at HUST in 2002 and 1994, respectively. He received his master degrees in the field of Pattern Recognition and Intelligent Control from HUST in June 1997. He had been a visiting scholar for one year at the Polytechnic School of Engineering of New York University since 2013. His interest focuses on multimedia networking. His current research includes P2P media streaming, mobile video networking and online recommendation for multimedia systems. He won the Science and Technology Progressive Award of Guangdong Province in 2011 and 2015, and the Sci. and Tech. Innovation Award of SARFT of China in 2012, respectively. He is a member of IEEE and ACM and a senior member of CCF.