Follow Me: Personalized IPTV Channel Switching Guide

Chenguang Yu
ECE
New York University
2 MetroTech Center
Brooklyn, NY 11201
chenguang.yu@nyu.edu

Hao Ding
ECE
New York University
2 MetroTech Center
Brooklyn, NY 11201
hao.ding@nyu.edu

Houwei Cao
CS
New York Institute of Technology
1855 Broadway
New York, NY 10023
hcao02@nyit.edu

Yong Liu
ECE
New York University
2 MetroTech Center
Brooklyn, NY 11201
yongliu@nyu.edu

Can Yang
CSE
South China University of Technology
Guangzhou 510006, China
cscyang@scut.edu.cn

ABSTRACT

Compared with the traditional television services, Internet Protocol TV (IPTV) can provide far more TV channels to end users. However, it may also make users feel confused even painful to find channels of their interests from a large number of them. In this paper, using a large IPTV trace, we analyze user channel-switching behaviors to understand when, why and how they switch channels. 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. Evaluation on the IPTV trace demonstrates that our recommender systems can achieve up to 45 percent hit ratio with only three candidate channels. Our recommender systems only need access to user channel watching sequences, and can be easily adopted by IPTV systems with low data and computation overheads.

KEYWORDS

IPTV, Recommender System, Channel Switching, Realtime Recommendation, Fusion Method

ACM Reference format:
Chenguang Yu, Hao Ding, Houwei Cao, Yong Liu, and Can Yang. 2017. Follow Me: Personalized IPTV Channel Switching Guide. In *Proceedings of MMSys'17, Taipei,Taiwan, June 20-23, 2017,* 11 pages.
https://doi.org/10.1145/3083187.3083194

1 INTRODUCTION

With the wide adoption of smart TV and Internet protocol TV (IPTV), TV users can now get much more channels and better watching experience. However, the long standing problem of *"which channel to watch?"* still bothers them even in the current IPTV era. How to guide users to quickly find channels of their interests is critical for IPTV service providers to prevent their customers from switching to Over-the-Top (OTT) content providers, such as Netflix or Hulu. Recommendation Systems (RSs) have been widely adopted by Internet-based content service providers, such as Amazon, Netflix, and Spotify, to enhance their customer experience. Advanced machine learning techniques, such as collaborative filtering, natural language processing, and multimedia content analytics, have been developed to automatically generate recommendations for books, movies, music, etc. IPTV providers have all the technical means to enhance their services by developing channel RS that generates personalized channel recommendation for each user. But we haven't seen much RS adoption in IPTV so far. Most TV services only offer Electronic Program Guide (EPG), which is a long list of channels organized in a multi-layer menu. General EPG is not tailored to individual user tastes. EPG is also annoying to end users – whenever a user wants to switch channel, she needs to navigate to EPG first, reads through the long channel list, then jumps to a channel of her interest. It is possible to generate *personalized EPG* for a user by augmenting the full channel list with a short list of channels that are most likely watched by the user, as predicted by the channel RS. 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.

Each channel consists of a sequence of programs. Some programs have fixed periodic schedules, e.g. daily news at 7pm, and content continuity, e.g. TV show series. Some programs are only broadcasted once, e.g., live sports events, and maybe at unpredictable time, e.g. coverage of emerging events. One way of recommending channels is to recommend programs inside channels. For example, if program A is scheduled to be broadcasted on channel x starting at time t, and we predict that user u will like program A, then we will recommend channel x to user u at time t. However, there are several challenges to this program-based channel RS. First of, in Collaborative Filtering (CF), to recommend an item to a user, we need to know the ratings of this item by other users similar

MMSys'17, June 20-23, 2017, Taipei,Taiwan

ACM ISBN 978-1-4503-5002-0/17/06…$15.00
https://doi.org/10.1145/3083187.3083194

to the target user. This works well for books, music, and movies, which users consume/rate asynchronously. In live IPTV channels, all users watch the same program at the same time, there is no rating history for the program to leverage on, i.e., the well-known "cold item" problem for CF. Content analysis is one way to deal with the "cold item" problem. By analyzing program metadata, such as category, description, producer, actors, etc., we can recommend to a user new programs that have content features similar to the programs liked/watched by the user before. To achieve this, we will have to maintain and analyze various metadata of old programs watched by users. We also need 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. Finally, since we are interested in realtime channel recommendation and users are free to leave and join a channel in the middle of a program broadcast, should we recommend a user to watch a program that is half-way through or close-to-end? All those complexities suggest that a channel is a much more volatile item for recommendation than a book or a movie, and program-based channel RS is not always feasible and efficient.

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. Since our RSs only work at the channel level, they are completely oblivious to the schedule and content diversity at the program level. They can be easily integrated into the existing IPTV systems without incurring much computation and data overhead. Towards this goal, using a rich trace of real IPTV users, we first conduct a thorough analysis on user channel switching behaviors with a focus on *when, why and how* they switch channels. We show that users have different needs of channel recommendation for different types of channel switching. Based on insights obtained from user behavior analysis, we then develop several base RSs that can already give good channel recommendations by exploring basic user and channel features, such as global and personal channel popularity, personal schedule, channel transition pattern, etc. We further improve the accuracy of base RSs through model fusion with different prediction models, ranking approaches, and data partitioning. Through evaluation on the real IPTV user 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 45% when only three channels are recommended.

The rest of the paper is organized as follows. In Section 2, we discuss the related work on RSs for TV channels. In Section 3, we analyze users' channel watching and switching patterns. Based on the statistic analysis, in Section 4, we propose our base and fusion RSs and evaluate their performance. Section 5 concludes the paper.

2 RELATED WORK

TV user behavior study was very limited in both scale and accuracy until IPTV made large-scare monitoring and survey possible. With the booming of OTT content, both content popularity and user behaviors are well studied, e.g, [1], [9] and [18]. This paved the way for the wide adoption of RS among OTT providers, such as Netflix, Spotify and YouTube. Treating TV channels as OTT content, Cha et al. [5] 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. [23] also conducted IPTV channel popularity analysis and focused on its temporal dynamics. Measurement and modeling of video watching time in a large-scale Internet video-on-demand system was presented in [9]. In [18], user behaviors for live and on-demand content were compared for an IPTV system delivering both types of content. Different from the previous measurement studies, we focus on user channel switching patterns that can be mined for realtime channel recommendation.

Accurate channel switching prediction can improve user experience in different ways. For example, studies in [15], [27] and [29] 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., [13, 14, 22], have been proposed to match users' personal interests with huge amount of content choices. In the TV domain, most RSs were built to address the program recommendation problem [26]. After the first EPG was introduced by Das et al. [11], a series of rule-based, statistical or machine learning approaches have been proposed for TV program RS, e.g., [2, 8, 12, 17, 24, 25, 28, 30]. Different from those studies, our RSs directly recommend channels, instead of programs. Compared with program RSs, channel RSs are more flexible, do not require extra program information, and can be adopted by IPTV systems with low data and computation overheads.

There are also several RSs for TV channels. Lee et al. [16] analyzed user channel watching behaviors in terms of recency and frequency and developed a channel RS. Park et al. [21] proposed a recommendation algorithm based on user channel switching history. Chang et al. [7] proposed a TV channel RS based on the feedback loser tree (FLT) algorithm. Chang et al. [6] and Oh et al. [20] developed TV channel RSs based on collaborative filtering methods, such as Matrix Factorization. Ning et at. [19] considered TV channel recommendation as a channel profit maximization problem: how to switch among n channels, each of which contains at most k live shows. Compared with the existing channel RSs, our improvements are two folds: firstly, we introduce six base RSs to generate channel features and achieve higher recommendation accuracy using fusion RS models; secondly, our system requires only user channel watching sequences and can generate channel recommendation on-the-fly to guide user channel-switching in realtime.

3 CHANNEL SWITCHING ANALYSIS

To define a realistic and relevant channel recommendation task for IPTV users, we first need to understand their channel switching behaviors and their needs for channel recommendation. More specifically, we want to know *when, why, and how* users switch channels, and how difficult they can find interesting channels.

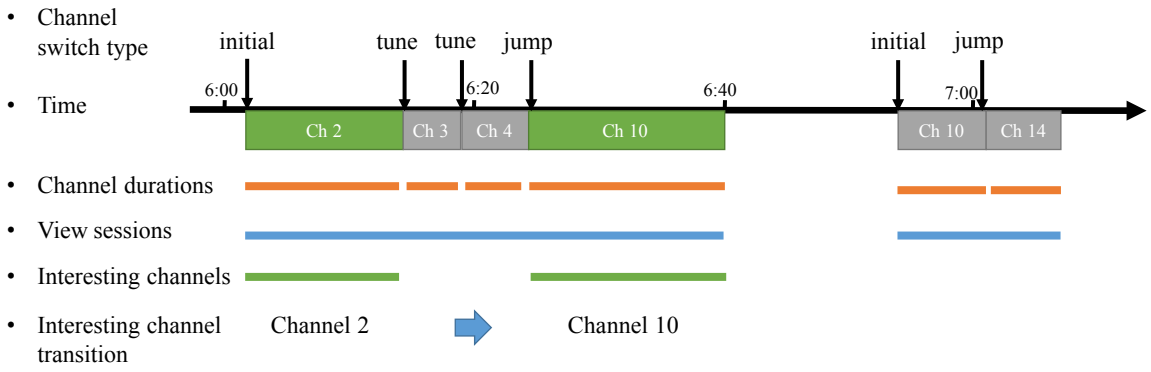


Figure 1: A sample of raw channel-watching log and the derived user behavior statistics.

3.1 Dataset and Terminology

Our dataset is provided by a major IPTV service provider for the metropolitan area of Guangzhou, China. It consists of user channel watching logs for the entire month of August 2014. Each log is a four-tuple:

{user_id, channel_id, start time, end time}

There are totally 222K users, 133 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 directly 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 analysis, 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 switchings in Figure 1 are labeled correspondently.
- **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 captured by any channel switching. As the example 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.

3.2 Channel Watching Statistics

3.2.1 *how many channels a user watches?*

As the first step of our user behavior study, we report in Table 1 the distribution of the number of channels watched by a user over different time periods (daily, weekly, monthly). Each cell has two numbers. The first one is the number of channels that the user watched for at least 10 minutes, the second one is the number of all channels watched by the user, regardless of the duration. We can make two interesting observations from Table 1. First of all, most users only watch a small subset of channels. For example, out of the 133 channels, the median number of channels watched by a user each day is only 7 and only 3 channels are watched for longer than 10 minutes. This suggests that users may need help to search for and find more interesting channels to watch. Meanwhile, the numbers of watched/interesting channels increase sub-linearly as we increase the time period from a day, to a week, then to a month. This suggests that users' interests are kind of consistent over time and a channel RS can predict a user's future interest by properly mining the user's channel watching history.

3.2.2 *when are channels watched?*

Next we investigate the temporal pattern of users' channel watching activities. In Figure 2 , we plot for each channel the heat-map of the channel-watching duration from all users over 4 weeks. Each color block encodes the aggregate channel-watching duration for each channel at each hour-of-day accumulated for 4 we‘ eks (e.g., the left bottom corner block represents the total watching duration

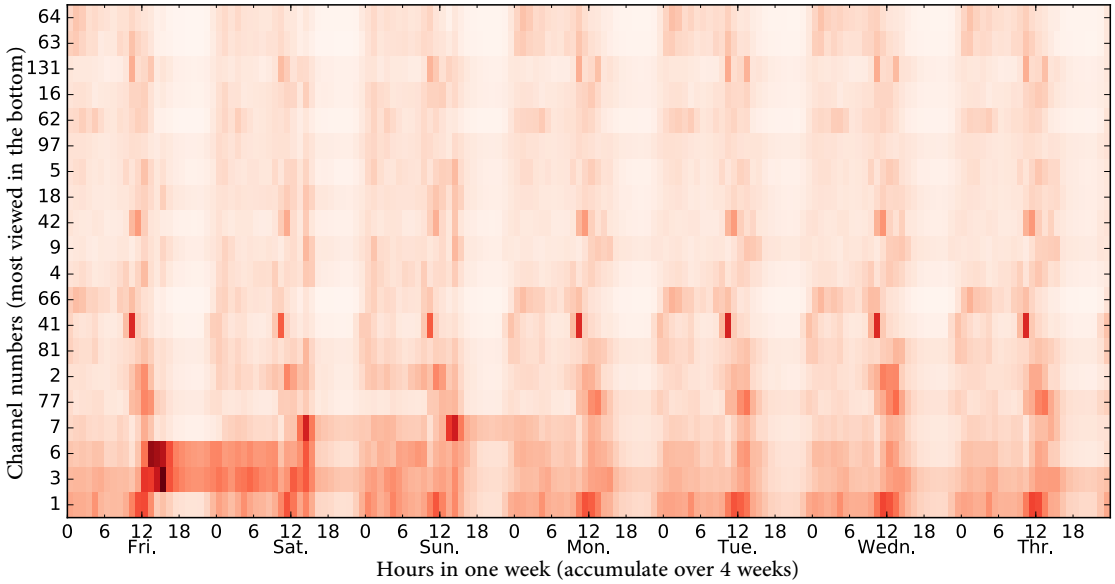


Figure 2: Heatmap of Channel Watching Activity: the color of the block for each channel at each hour-of-week encodes the aggregate watching time from all users over four weeks.

Table 1: The number of unique channels watched by a user. The first number in each cell is for channels watched for at least 10 minutes and the second number is for all channels watched.

Percentile	Monthly	Weekly	Daily
10%	4/10	2/4	1/1
25%	8/21	3/8	1/3
50%	14/39	6/17	3/7
75%	22/62	10/30	4/13
90%	30/85	15/47	6/21

of Channel 1 between UTC 0 am - 1 am for the four Fridays in our trace). Clearly, there is a strong weekly pattern in Figure 2.

For example, Channel 41, a local news channel, has an obvious peak hour everyday, during which it always attracts more users than other channels (vertical comparison). For the same channel, the aggregate watching duration in each peak hour is also significantly higher than those in the other hours (horizontal comparison). Users also tend to concentrate on a few channels (e.g., weekly TV shows of Channel 3, 6) on Friday night. Motived by this temporal pattern, we will include the hourly channel watching schedule as an important feature for our proposed channel RSs in Section 4.

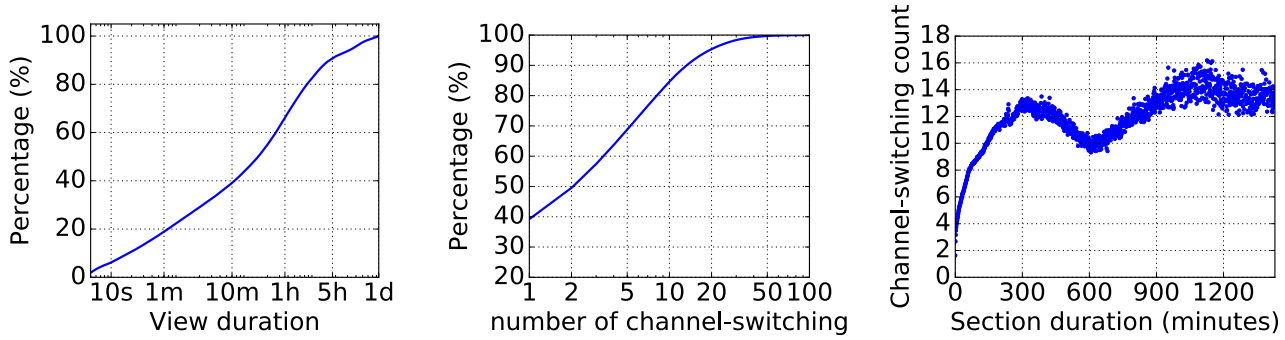
3.3 Channel Switching Statistics

3.3.1 how frequent channel switching is?

Figure 3a is the Cumulative Distribution Function (CDF) plot of the duration of watching sessions. More than half of the sessions are from 10 minutes to 5 hours, and the median is around 24 minutes. Figure 3b plots the distribution of the number of channel switchings (i.e., the number of channels watched minus one) within a session. 40% of the view sessions don't have any channel switching (number of switch = 1). Meanwhile, there are more than 25% of sessions have five or more channel switchings. Figure 3c is the scatter plot of channel switching counts vs. session durations. Durations are binned for each minute, and the average channel switching count for sessions in each bin is plotted. The switching count increases as the duration increases till around 300 minutes, but decreases between 300 minutes and 600 minutes. Some of those long watching-sessions are probably inactive sessions, e.g., users may just let the TV on and don't watch it actively.

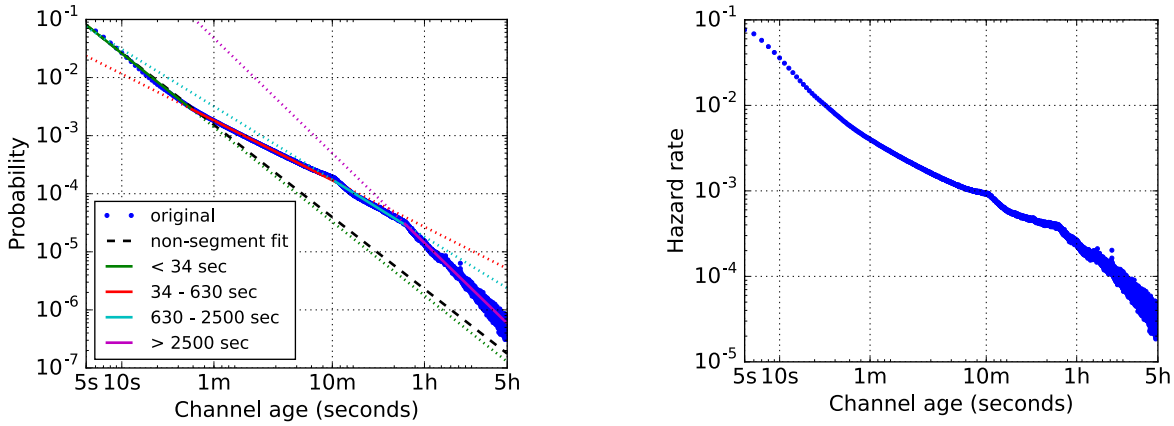
3.3.2 how long a user stays with a channel?

Channel watching duration - the duration that a user stays with a channel is a simple and effective measure of how interesting the channel is to her. Now we analyze how long a user watches a channel before her interest fades away. In Figure 4a, we plot the probability mass function of channel-watching duration in the log-log scale. The piece-wise linear curve suggests that the channel-watching duration generally follows the power law. The black dot line in Figure 4a is the over-all fitting line based on power law. It performs well before one minute and shifts away after that. To increase the accuracy, we fit the duration distribution with a segmented power law distribution, with four segments: $[0, 34)$, $[34, 630)$, $[630, 2500)$ and $[2500, \infty)$. The segmented curve matches the original curve better.The Kolmogorov-Smirnov test score drops from 0.233 to 0.007. The fact that power law exponent varies with watching duration suggests that a duration-dependent channel switch-out rate, which reflects different channel switching behaviors at different stages of channel watching. We define *channel switching out* rate as the



(a) Duration (b) Switching Count (c) Count vs. Duration

Figure 3: Statistics of Channel Watching Sessions



(a) Duration Distribution. (b) Switching Out Rate.

Figure 4: Channel Watching Duration and Switching Out Rate.

hazard rate λ used in survival analysis, which is the probability of a user switches out of a channel immediately after her watching duration reaches t conditional on she has watched the channel for longer than t:

$$\lambda(t) \triangleq \lim_{\Delta t \to 0} \frac{P(t \leqslant T < t + \Delta t)}{\Delta t \cdot P(T > t)},$$

where T is a random variable denoting a user's channel watching duration.

We plot the channel switching out rate in Figure 4b. As expected, the channel switching rate (hazard rate) decreases as channel watching duration grows. However, we also notice two turning points at the curve near 10 minutes and 40 minutes. The fact that channel switching out rate decreases faster at the turning points suggest after a user watches a channel for more than 10 minutes or 40 minutes, she becomes significantly more stable and tends to watch the same channel longer. In other words, watching durations of 10 minutes/40 minutes can be used as thresholds to judge a user's short-term/long-term interest on a channel.

3.3.3 *how users switch channels?*

As defined earlier, there are three ways through which a user can switch to a channel: **Initial**, **Jump**, and **Tune**. Table 2 lists the counts and percentages of each of them. It also lists the mean and median duration of channel watching time immediately after each type of switching.

It is obvious that how a user switches to a channel has a strong impact on how long she will watch the channel. As further shown in Figure 5a, "jump-to" channels are watched significantly longer then "tune-to" channels. This can be explained as users have some pre-knowledge or expectation on what channels may interest them when they decide to input channel numbers on their remotes directly. Meanwhile, when a user does channel surfing using the "tune" method, she normally does not have a clear preference on which channel to watch. Consequently, she will quickly go through

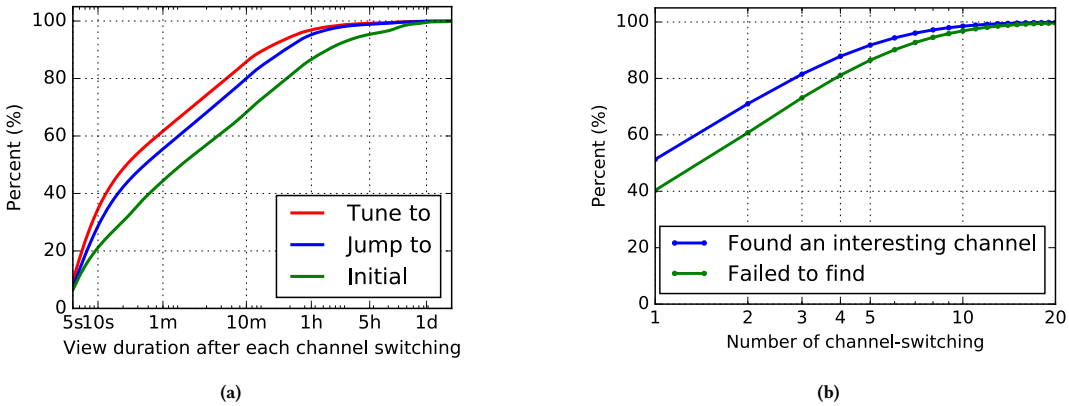


Figure 5: Statistics of Channel Switching: a) watching duration after each channel switching; b) the number of switchings for each channel searching.

Table 2: Statistics of different types of channel-switching

Switch Type	Count (Million)	Ratio (%)	Channel Watching Duration (Sec.)	
			Mean	Median
Jump	43.43	61.02	1142	37.4
Tune	16.06	22.57	812	23.0
Initial	11.68	16.41	3685	100.6

a sequence of channels before settling down on an interesting channel. Finally, "initial" channels are watched the longest among the three. The reason might be that some users configure their TV and set up their most favorite channel as the default channel to show after power-on. It might also be due to the TV is not being watched or controlled by an active user after power-on, e.g., TVs in public places, such as bars and restaurants.

It takes users time and efforts to find interesting channels to watch. Naturally, we want to know whether a user can always find interesting channels, and if so, how many channel switchings she has to do. We call the sequence of channel switching activities between the end of last "interesting" channel or power-on, and the next "interesting" channel or power-off as "channel searching". Among all channel searching activities, users fail to find the next interesting channel 28% of time. Among the rest 72% successful channel searching activities, 43% interesting channels are found by **Jump**, 13% by **Tune**, and 16% by **Initial**. As illustrated in Figure 5b, if a user ever finds an interesting channel, she needs to switch channel more than once with 50% chance, and more than 5 times with 10% chance. It takes even more channel switchings before a user gives up without finding any interesting channel.

4 CHANNEL RECOMMENDER SYSTEM

The analysis in the previous section demonstrates that users either fail to find interesting channels to watch or have to switch channels multiple times before reaching an interesting channel. In this section, we develop realtime channel RSs that automatically recommend a list of channels to a user whenever she wants to switch channel. The goal is to help users quickly find interesting channels to watch.

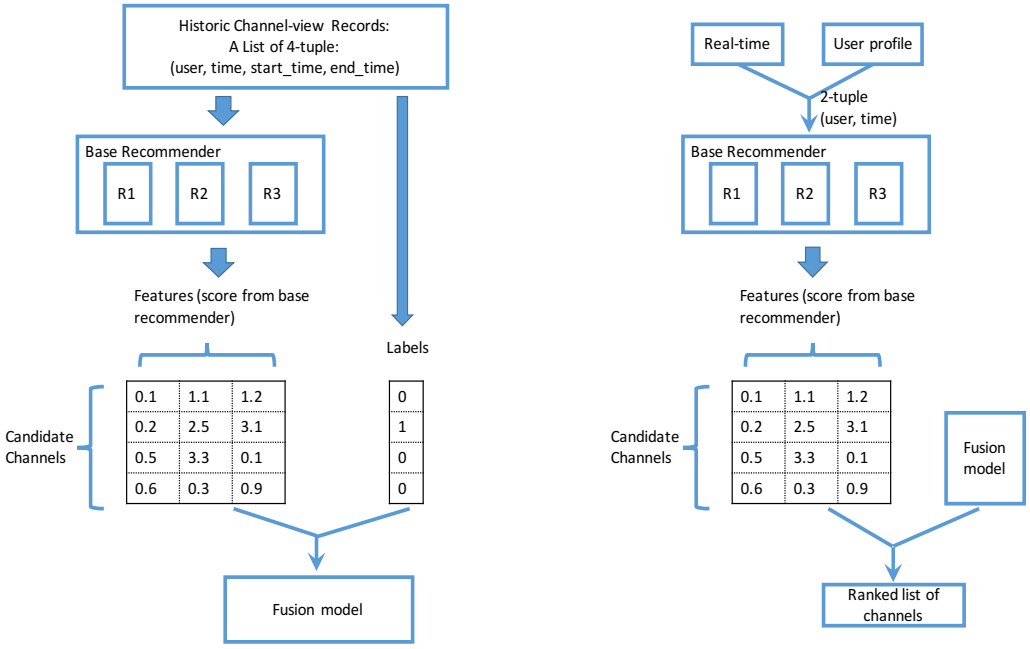
4.1 RS Task and Workflow

As we discussed in Section 3.3.3, channel-switching is triggered by user's desire of seeking a new interesting channel. Although channel RSs can generate real-time recommendation at any time, we choose user-initiated channel-switching actions to be the recommendation trigger. For the example in Figure 1, recommendations 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 past channel-watching logs of a large number of users, for each target user u, 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 6: *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 a fusion RS model as will be described in Section 4.4. During the recommendation phase, we use (user, time) as the only input, and feed it to trained base RSs to generate features, which are fed to the trained fusion model to generate realtime channel recommendations.

4.2 Base RSs

Our recommendation system is built on several well-known recommendation methods. Some of the methods were also used in [25]. The results of those methods will be utilized as features for fusion RSs. This section will first describe all six base methods we used and then compare their recommendation accuracies.



(a) Training Phase **(b) Recommendation Phase**

Figure 6: Work Flows for the Proposed Realtime Channel Recommender System

Basic User-Channel Relation Features From channel-watching logs, we can easily derive various user-channel relation features:

- $\mathcal{U}(c,t)$: 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 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)

Current Global Popularity This method recommends the most popular channels among all users at any given moment t, i.e.,the channels currently watched by most users. The score of each channel c for user u at time t is:

$$score^{gp}_{c,u,t} = |\mathcal{U}(c,t)|,$$

where $|\cdot|$ denotes set size. This score is independent of users.

Historical Personal Popularity This method recommends the channels watched 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).

Personal Schedule This method recommends channels based on the user's channel watching history at specific time slots within a history window. The score of 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, for $t = 8:14pm$, it belongs to the $[8pm, 9pm)$ hourly slot. The channels watched most by a user between $8pm$ and $9pm$ in the past month will be recommended to the user. We choose one hour as the time slot length in our experiments.

User-based Collaborative Filtering This method recommends channels that are watched by most similar users, a.k.a. 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 $\mathbf{D}_u^{\mathcal{T}}$:

$$\mathbf{D}_u^{\mathcal{T}}[c] = d(u,c,\mathcal{T}), \quad \forall c \in \mathbf{C}.$$

The similarity $sim(u,v,t)$ between user u and v at time t is the cosine similarity of their channel-watching duration vector $\mathbf{D}_u^{[t-\Delta,t)}$ and $\mathbf{D}_v^{[t-\Delta,t)}$ during the window Δ before t:

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

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

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

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:

$$p_u(c|c') = \frac{|S_u(c' \to c)|}{|S_u(c' \to *)|},$$

where $S_u(c' \to c)$ is the set of user u's interesting channel switching actions from c' to c, and $S_u(c' \to *)$ is the set of user u's interesting channel switching actions from c' to any other channel. The recommendation score of each channel c and user u at time t 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.

LDA Topic Model This method recommends channels based on Latent Dirichlet Allocation (LDA) topic model [3]. In LDA, a collection of words (i.e., a document) may be explained as a mixture of several latent topics. The probability that a word appears in a document can be estimated by the topic distribution of the document and the word distribution of a topic.

We treat a user's channel-viewing log as a "document", which consists of a sequence of "words", each of which is a 2-tuple (channel, hour-of-day). The intuition behind this is: even though we don't have program information for each channel, since many channels schedule repetitive programs or programs with a similar type in the same hourly slot of a day, such as daily news program at 7pm, or football matches at 1pm, each (channel, hour) tuple can be treated as a repetitive program or programs with similar types.

If a LDA model is trained based on "documents" of user channel viewing logs, we can predict the most possible "words" (compounds of a channel and a hour) for each user. More specifically, the LDA method predicts latent topic/interest distribution of each user (represented by the "document" of her channel viewing log) and the latent topic/interest distribution for each (channel, hour) tuple i.e, the "word" for LDA. Finally, the most possible "word" (X,Y) for user u can be easily interpreted as user u is most likely to watch channel X in hour Y. Terms used in our LDA model are defined as following:

- **Pseudo word** $w(c,h)$: a 2-tuple of (channel, hour) treated as a virtual program,
- **Word frequency** $f(c,h)$ **in document** d: the summation of watching duration of a pseudo word $w(c,h)$ by a user d within a time window,
- **Topic** k **of user** d: it reflects a latent topic of a user; a user may have multiple possible latent topics,
- **User-topic distribution** $\theta_{u,k}$: probability of a user u contains a topic k,
- **Topic-(channel, hour) distribution** $\lambda_{k,w(c,h)}$: the probability of a topic k contains a channel-hour tuple $w(c,h)$.

Finally, the recommendation score of channel c for user u at time t is defined as:

$$score^{lda}_{c,u,t} = \sum_k \theta_{u,k}\lambda_{k,w(c,h(t))},$$

where $h(t)$ is the hour-of-day that t belongs to.

4.3 Base RS Performance.

Training/Testing Separation. Our 31-day dataset is split into a training set (the first 24 days) and a testing set (the last 7 days). The training set is used for training (LDA method) or statistical analysis (all the other base methods). The testing set is used to test the accuracy of base RSs. **Experiment Metric: top-k hit ratio.** Whenever a user initiates a channel switch, our system will generate a short list of k channels out of more than 100 channels. We call a top-k channel list a *hit* if one out of the k recommended channels is indeed watched continuously by the user for at least 10 minutes within next 20 minutes interval. The *top-k hit ratio* for a RS is simply the fraction of hits among top k-channel lists generated by the RS for all channel switches. As in Figure 7, all other methods outperform "global popularity" method in most cases. "personal popularity" and "personal schedule" methods are the best among them. *LDA topic model has poor performance, this may due to the overly-simplified assumption we made: the same program is always broadcast at the same hour of a day.*

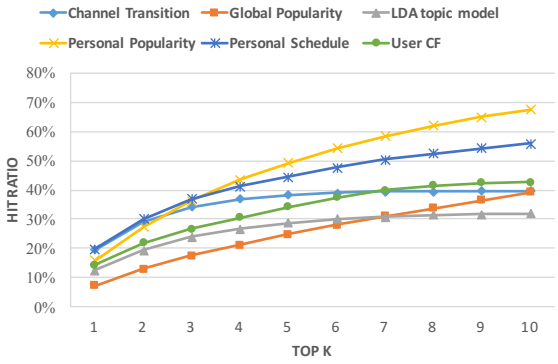


Figure 7: Top-k Hit Ratio of Base Channel Recommender Systems

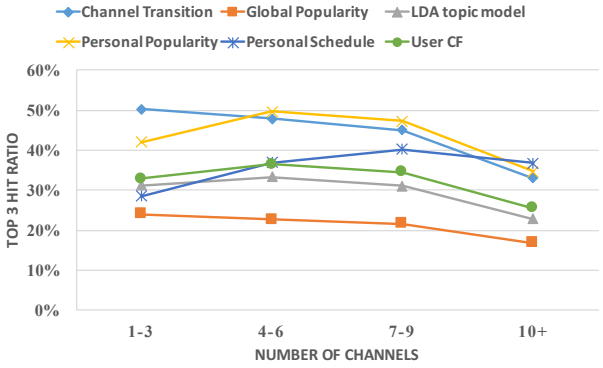


Figure 8: Top-3 Hit Ratio of Base Channel Recommender Systems for Users with Different Watching Activity Levels.

In Figure 8, we further compare the RS performance (Top-3 hit ratio) for users with different watching activity levels, which are measured by the number of channels that are watched by a user for more than 10 minutes during one week. For users who watched no more than three channels in a week, channel transition RS works the best. This is because users only switch between a small

number of candidate channels. But for users who watched more channels, the channel transition RS performance is not as good, simply because now there are more channels that a user can switch to. Personal popularity RS works the best for users who watched 4 to 9 channels. This suggests that for this group of users, their interests have some consistence over time. Finally, for active users who watched more than 10 channels, the personal schedule RS is slightly better than personal popularity. This is because those users tend to spend longer time watching TV, and have different interests in different time slots.

Finally, Figure 9 plots the distribution of (user, week) tuples and recommendation events over different watching activity levels. While the numbers of (user, week) tuples falling into all levels are comparable, vast majority of recommendations are generated for active users, since they initiate more channel switches. As a result, the average hit-ratio (averaged over recommendation events, not over users) in Figure 7 is dominated by hit ratios for active users, for whom the base RSs do not work as well as less active users. Motivated by this, we will study fusion algorithms to further improve base RS accuracy, especially for active users.

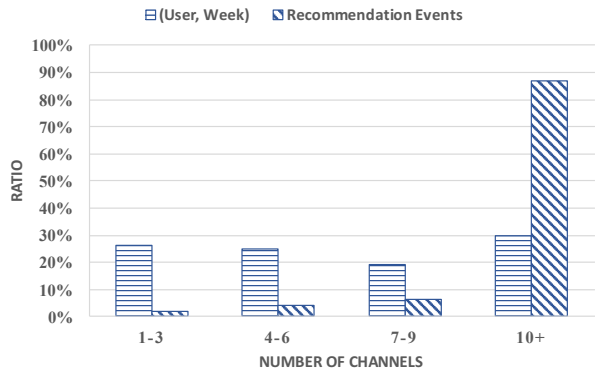


Figure 9: Distribution of Users and Recommendation Events over Different Watching Activity Levels.

4.4 Fusion Recommender Systems

Now we study fusion RSs that combine scores generated by base RSs using some fusion function $\mathcal{F}(\cdot)$ to produce the final scores for all channels and recommendation list.

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

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

Prediction Models. The goal of a fusion RS is to predict whether a channel will interest a user based on the scores obtained by the 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), Support Vector Machine (SVM)* [10] and *Random Forest (RF)* [4]. 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, or RF*), which will be used to generate fusion scores for candidate channels, and consequently recommendation lists in the test phase.

Ranking Approaches. We investigates 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_{chi} is the score vector of channel i and $true/false$ is the label of whether the user watched channel i. 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 ranks higher than channel B, it will get one vote. Finally, channels with most votes will be placed at the top of recommendation list.

Data Partition. We also need to consider whether and how our data should be grouped for model training. For example, on one hand, if we train one 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/hours and train multiple 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).

4.5 Fusion RS Performance

To search for the best fusion setting, we enumerate all combinations of the previous three 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, and data from the last 7 days are used for testing. In Table 3, we compare the recommendation accuracy regarding 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 3 due to their bad performance. Among all other settings, we can see random forest (RF) models outperform other models, and the setting of per-hour pairwise random forest model is the best among all of them. It achieves significant improvement over the individual base RSs.

Table 3: Performance Comparison of Fusion Channel RSs.

	Best Single	Score Sum	LR	RF	LR per-hour	RF per-hour	LR Pairwise	RF Pairwise	RF Pairwise per-hour
Top 1	19.8%	19.2%	21.4%	22.4%	21.4%	22.9%	21.0%	22.8%	**23.3%**
Top 3	36.9%	39.8%	42.1%	43.9%	42.2%	44.5%	42.0%	44.5%	**45.0%**
Top 5	49.3%	51.7%	53.3%	55.6%	53.5%	56.0%	53.6%	55.8%	**56.6%**

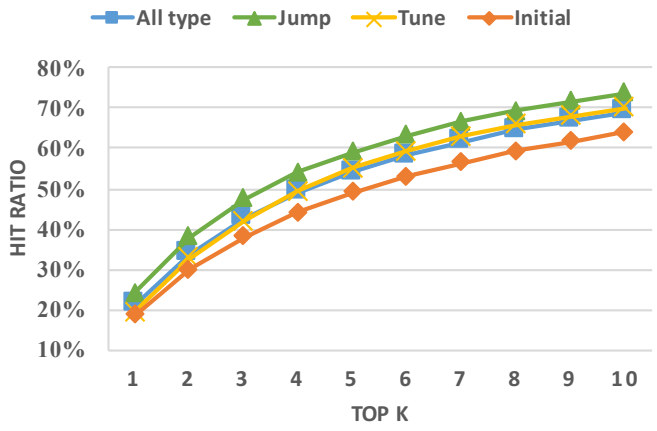


Figure 10: Top-k Hit Ratio of Fusion Channel RS for Different Switching Types.

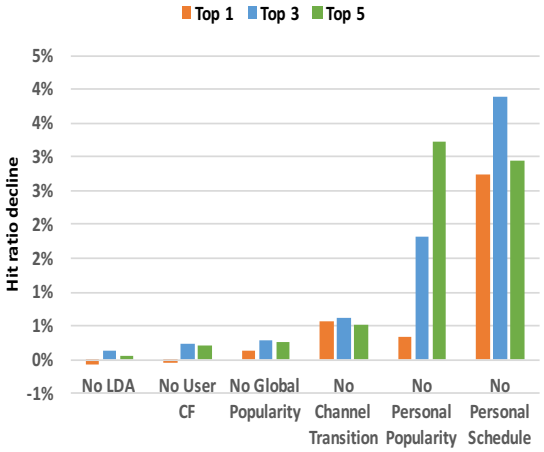


Figure 11: Performance of Fusion RS after Removing One Base RS.

We also found that our RSs generally perform better for "jump" type of channel switching as illustrated in Figure 10. 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" switching, since this is when a user is less clear about what she should watch.*

To evaluate whether RSs can provide complementary information, *leave-one-out* models are generated and tested. We pick one base RS at a time and compare the performance of fusion RS without it. As in Figure 11, all base RSs can improve the final performance (except some top-1 cases), and personal popularity and personal schedule methods are the most important two. This is consistent with performance of individual RSs in Figure 7.

Meanwhile, we should also notice that more computational expensive methods, such as collaborative filtering and LDA model, are not necessarily more important base RSs contributing to the accuracy of our final fusion RS. Some straightforward methods, such as personal popularity and personal schedule, may already catch most import features of our dataset and have decent performance on their own. Our study demonstrated the importance of statistical analysis on data patterns prior to RS design. By carefully developing and fusing different base RSs, we were able to strike the desired balance between the computation overhead and recommendation accuracy in the fusion RS.

5 CONCLUSIONS

In this paper, we studied realtime channel switching recommendation for IPTV systems. Using a large IPTV user trace, we first conducted a thorough user study and gained valuable insights on when, why and how they switch channels. We then developed six base RSs that generate channel recommendations using basic user-channel features, such as personal schedule, personal channel popularity, channel transition patterns, as well as classic recommendation methods, such as global popularity, user-based CF, and LDA. We further improve the accuracies of base RSs using different fusion methods. Through extensive evaluation, we demonstrated that our fusion RS model outperforms individual base RSs and can accurately guide user channel switching by using extremely short recommendation lists. Our proposed RSs incur low data and computation overheads, and are suitable for realtime channel recommendation in practical IPTV systems.

6 ACKNOWLEDGEMENT

This project was partially supported by the National Natural Science of Foundation of China under contract *U*1611461 and the Oversea Collaboration Project #201704030124, funded by Guangzhou City, Guangdong Province, China.

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*, pages 199–210, 2012.

[2] 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*, pages 3–26. Springer, 2004.
[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.
[4] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
[5] 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*, pages 71–84, 2008.
[6] H.-Y. Chang, S.-C. Huang, and C.-C. Lai. A personalized iptv channel-recommendation mechanism based on the mapreduce framework. *The Journal of Supercomputing*, 69(1):225–247, 2014.
[7] 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*, 57(12):1776–1789, 2014.
[8] 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*, volume 2, pages 260–263. IEEE, 2012.
[9] 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 Trans. on*, 15(8):2087–98, 2013.
[10] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
[11] D. Das and H. ter Horst. Recommender systems for tv. In *Recommender Systems, Papers from the 1998 Workshop, Technical Report WS-98-08*, pages 35–36, 1998.
[12] 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*, pages 166–174. Springer, 2007.
[13] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from noisy entries. *Journal of Machine Learning Research*, 11(Jul):2057–2078, 2010.
[14] Y. Koren, R. Bell, C. Volinsky, et al. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.
[15] E. Lee, J. Ku, and H. Bahn. An efficient hot channel identification scheme for iptv channel navigation. *Consumer Electronics, IEEE Trans. on*, 60(1):124–9, 2014.
[16] H. Lee, S. Lee, H. Kim, and H. Bahn. Personalized recommendation schemes for dtv channel selectors. *IEEE Transactions on Consumer Electronics*, 52(3):1064–1068, 2006.
[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*, pages 430–435. IEEE, 2004.
[18] 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 Trans. on Multimedia Computing, Communications, and Applications (TOMM)*, 10(3):23, 2014.
[19] L. Ning, Z. Zhao, R. Zhou, Y. Zhang, and S. Feng. Realtime channel recommendation: Switch smartly while watching tv. In *International Workshop on Frontiers in Algorithmics*, pages 183–193. Springer, 2016.
[20] S. Oh, N.-r. Kim, J. Lee, and J.-H. Lee. Comparison of techniques for time aware tv channel recommendation. In *Soft computing and intelligent systems (SCIS), 2014 joint 7th international conference on and advanced intelligent systems (isis), 15th international symposium on*, pages 989–992. IEEE, 2014.
[21] S. Park, S. Kang, and Y.-K. Kim. A channel recommendation system in mobile environment. *IEEE transactions on consumer electronics*, 52(1):33–39, 2006.
[22] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8, 2007.
[23] 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*, volume 37, pages 275–286, 2009.
[24] B. Smyth and P. Cotter. Surfing the digital wave: Generating personalised {TV} listings using collaborative. *Case-Based Recommendation, LNCS*, 1650, 1999.
[25] 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.
[26] 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*, 42(22):9046–9076, 2015.
[27] C. Yang and Y. Liu. On achieving short channel switching delay and playback lag in ip-based tv systems. *Multimedia, IEEE Trans. on*, 17(7):1096–1106, 2015.
[28] 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), IEEE international conference on*, pages 1–6, 2014.
[29] S. Zare and A. G. Rahbar. Program-driven approach to reduce latency during surfing periods in iptv networks. *Multimedia Tools and Applications*, pages 1–13, 2015.
[30] J. Zhang, Y. Li, M. Chen, and L. You. An implicit feedback integrated lda-based topic model for iptv program recommendation. In *Communications and Information Technologies (ISCIT), 2016 16th International Symposium on*, pages 216–220. IEEE, 2016.