Watch-It-Next: A Contextual TV Recommendation System

Michal Aharon[1], Eshcar Hillel[1], Amit Kagian[1], Ronny Lempel[2], Hayim Makabee, and Raz Nissim[1]

[1] Yahoo Labs, Haifa, Israel,
michala,eshcar,akagian,raz@yahoo-inc.com,
[2] Outbrain, Netanya, Israel

Abstract. As consumers of television are presented with a plethora of available programming, improving recommender systems in this domain is becoming increasingly important. Televisions sets, though, are often shared by multiple users whose tastes may greatly vary. Recommendation systems are challenged by this setting, since viewing data is typically collected and modeled per *device*, aggregating over its users and obscuring their individual tastes.
This paper tackles the challenge of TV recommendation, specifically aiming to provide recommendations for the next program to watch following the currently watched program on the device. We present an empirical evaluation of several recommendation methods over large-scale, real-life TV viewership data. Our extentions of common state-of-the-art recommendation methods, exploiting the current watching context, demonstrate a significant improvement in recommendation quality.

1 Introduction

In recent years, online experiences have made increasing use of recommendation technology and user modeling techniques. Media sites recommend what to read, music streaming sites recommend what to listen to, VOD subscription services recommend what to watch, and more. In many of these instances, recommendation effectively transforms an online service to be personalized - tailored to its single or primary user. This occurs when recommendations are exposed in an experience consumed through a personal device such as a smartphone or a laptop, or when they are exposed on personal accounts such as a social network account. However, in other cases recommendation technology is applied in settings where multiple users share an application or device. Examples include game consoles or high end smart TVs in household living rooms, family accounts in VOD subscription services or eCommerce sites, and shared desktops or tablets in homes. These multi-user cases represent a challenge to recommender systems, as recommendations given to one user may actually be more suitable for another user. In general, the utility of recommendations in multi-user settings is lower, sometimes to the point of frustrating users when the recommendations they receive are dominated by items suitable for others.

This paper tackles the TV recommendation problem by applying context to implicitly disambiguate the user, or users, that are watching it. Specifically, we address the household smart TV situation, where a television set runs software that identifies what is watched on it, and taps that knowledge to recommend to its owners what they should watch at any given time. Obviously, individual household members may have very different viewing habits and tastes. Furthermore, when certain combinations of household members watch TV together, they may watch programs that differ from what any of them would have watched alone. Thus, every combination of viewers at any time may require different recommendations.

A key observation we tap is that often viewers watch multiple programs in succession, in what we call a TV *viewing session*. Thus, a given watched show influences our prior belief over the show that might be watched next. The data at our disposal does not identify which household members were watching television at any point in time, hence we measure the accuracy of predicting the next watched show – as is common in offline evaluation of recommendation algorithms, we associate improved prediction with better recommendation.

Our methods decouple the recommendation from the learning algorithm that produces the model. Essentially, we *exploit models produced by context-less learning* in order to produce contextual recommendations. This allows our recommender to easily work with any learning component that maps users and items to low-dimensional vectors, and to benefit from any optimization improving this component. Specifically, trained models leverage the currently watched show as context for recommending the next show to watch, *without changing how modelling is performed*. To determine the affinity of the individual(s) watching the device and a show, we perform a simple 3-*way product* of the vectors representing the device, the currently watched show (context), and the shows to potentially watch next (items). The context enables the algorithm to recommend significantly more relevant watching options than those output by state-of-the-art context-less recommenders.

We implemented a recommender system, *WatchItNext*, that demonstrates the usefullness of our methods on top of two learning models: Latent Factor Modeling and Latent Dirichlet Allocation. We provide an empirical analysis of *WatchItNext* on two recommendation scenarios: (1) *exploratory setting* focusing on recommending items which have yet to be consumed by a user (e.g., for a recommended TV series, no episode has been previously watched[3]), and (2) *habitual setting* where recommendations also include items that were previously watched on the device (such as previous episodes of a series). Using large-scale real-life offline viewing data, which processed 4-months long viewing histories of 340,000 devices[4], we demonstrate improvements over these two TV recommendation scenarios. Our personalized and contextual recommendation methods significantly

[3] We model viewing patterns at the series level, not at the episode level.

[4] We are working towards getting this data published under the umbrela of the Yahoo! Webscope Program http://webscope.sandbox.yahoo.com.

outperform personalized non-contextual as well as contextual non-personalized baselines.

The rest of this paper is organized as follows. Section 2 surveys background and related work. Section 3 presents the algorithms used to tackle the problem. Section 4 details the experimental setup and Section 5 reports our experimental results. We conclude in Section 6.

2 Background

This section describes two methods that serve as building blocks in our algorithms section, followed by a review of previous work related to the TV and in general, the multi-user device recommendation problem.

2.1 Building Blocks

Latent Factor Model (LFM) Collaborative Filtering methods analyze relationships and interactions between users and items according to consumption patterns or user ratings. Latent Factor Modeling is a common approach in this field [19]. Following the assumption that the underlying behavioral pattern is low-rank, LFM approximates a user-item relationship matrix as a multiplication of two low rank matrices, one representing the users and the other representing the items. Therefore, a (latent) vector $\bar{u} \in \mathbb{R}^n$ is assigned for each user, and another vector $\bar{i} \in \mathbb{R}^n$ for each item. The recommendation score for user u and item i is computed by the inner product of the corresponding vectors, $\langle \bar{u}, \bar{i} \rangle$. Often, the latent vectors are modeled using an iterative optimization process, such as gradient descent.

Latent Dirichlet Allocation (LDA) *Latent Dirichlet Allocation* is a generative probabilistic model of a corpus, originally proposed for modeling text documents [9]. The intuition behind this model is that documents exhibit multiple topics, which are distributions over a fixed vocabulary W. The LDA algorithm performs data analysis to compute an approximation of the conditional distributions of the hidden variables (*topics*), given the observed variables (words of the documents). In its standard application, LDA gets as input a set of documents, along with their bag-of-words representation, often represented as a sparse matrix in which the (i, d) entry is the number of times the word i appeared in the document d. It produces a set of n topics, where each topic is a multinomial distribution vector in $R^{|W|}$ representing the probability of each word to appear in a document discussing that topic. The documents' latent topics can later be inferred according to the words they contain.

2.2 Related Work

Standard recommender systems log viewing history per device. A possible approach to enhance personalization in multi-user devices is using *context-aware* and in particular *time-aware* recommender systems. In the smart TV use case,

this is backed by the assumption that each member of the household has regular time slots in which she watches TV. For example, children are more likely to watch TV in the afternoon than late at night.

We review the general context-aware approach, existing video and TV content recommendation systems, and some papers that handle the multi-user problem in other domains, and compare these solutions with ours.

Context-Aware Recommendations Context-Aware Recommender Systems (CARS) exploit information on the situation in which the user is interacting with the system to improve rating predictions and provide more relevant personalized recommendations. Adomavicius et al. [1, 2] distinguish three main types of CARS: (1) those based on *pre-filtering*, which select the data that will be modeled according to the target recommendation context, (2) those based on *post-filtering*, which adjust the recommendation, e.g., by filtering or weighting the recommendations according to the target context, and (3) those based on *contextual modeling*, which incorporate contextual information into the model itself. Both pre- and post-filtering have the advantage of using any non-contextual algorithm while providing context-aware recommendations.

Pre-filtering techniques such as *item splitting* [5] and *micro-profiling* [3, 21] generate contextual models based only on the relevant items or users data, which may be sparse. These techniques also require knowing the explicit context, while in some cases the context can only be derived implicitly.

Many works devise new contextual modeling algorithms. Some are for general purpose, such as *tensor factorization* [17, 22], factorization machines [20], and context-aware factorization [4, 14, 16]. Others specifically aim at improving music recommendations [13] by extending the LDA model, improving Web search ranking via incorporating context into factorized models [23, 25] or learning a variable length Hidden Markov Model [10]. These algorithms are often complex, introduce a large number of model parameters to be learned, and are mostly not available as off-the-shelf solutions.

Our work adopts a two-phase post-filtering approach. We compute an initial score while considering the current item context, and then weigh this score using the time of day as an additional context. The advantage of our approach is that it allows us to leverage available contextual information while using a standard non-contextual model.

Time-aware recommender system utilze the time as the main context of the recommedation. The main disadvantage of this approach is that it delineates a *static* mapping between users or groups of users, which we refer to as *entities*, and time slots. This is often not a valid assumption, as people may switch preferred times for watching TV. Our proposed framework supports a *dynamic* approach, which – in addition to the time of day – also relies on the currently watched show as context for recommending the subsequent show to watch. Our main assumption is that the currently watched show can implicitly serve as a dynamic entity-proxy. For example, consider a morning in a household where either a child or an adult is watching the TV. Whether the current show is *"Good Morning*

America" or *"Dora the Explorer"* should affect the recommendation of what to watch next.

Xu et al. [26] uses LDA topic modeling in order to extract the browsing context of users. Their recommendation algorithm differs from ours as it only considers the similarity between the active browsing session (item context) and browsing access patterns (items topics) without considering the model of the active user.

Video and TV Recommendations Video and TV content recommender systems tackle the multi-user problem in different ways. Some systems, like [6] ignore this problem, others [15, 27] recognize that multi-user devices present a challenge for personalized recommendations, but do not try to address this issue. Netflix [11] and Android [18] suggest actively switching between users of a single device, other TV content recommnders [7, 30] also ask users to log-in and collect their explicit feedback. Although it is technically possible for users to identify themselves by signing into the device, very few users currently log-in while watching TV.

Some recommendation systems for interactive TV platforms [29] utilize time as context by performing tensor factorization on user-item-context tensor. In contrast, our algorithm only trains simpler, non-contextual models, and applies both temporal and sequential context when serving the recommendation.

YouTube also recomends users which video to watch next [8,12]. The problem of video recommendation, however, is different as the inventory is not limited, and more importanly the input is less noisy, since users must be active in choosing the video, and the system can leverage explicit feedback by the users.

Recommending to a Multi-User Device Television sets are a prime example of multi-user devices, however this setting is tackled also in other domains.

The work of Zhang et al. [28] focuses on the identification of users sharing an account. They developed a model of composite accounts as a union of linear subspaces, and then applied linear subspace clustering algorithms to identify the users. They also defined a statistical measure of the *compositeness* of an account, which is similar to our concept of device entropy (see Section 5.1). While their model is built on top of explicit ratings provided by users, our work is based on implicit signals.

A recent work by White et al. [24] explores personalized web-search on shared devices. They present methods for identifying shared devices, estimating the number of searchers on each device, and attributing new queries to individual searchers. The estimated number is used to guide a *k-means clustering* in segregating the device search log into logs of k searchers. A new query is assigned to a cluster by applying a similarity score that is based on features such as topic, time, and query length.

Recommender	Description
$GeneralPop$	General popularity of item i
$TemporalPop$	Popularity of i at time-of-day t
$SequentialPop$	Popularity of i watched after c
$DevicePop$	Popularity of i within device d
$DevicePop + X$	$DevicePop$ combined with a recommender X
LFM	Latent Factor Model with stochastic gradient descent
LDA	Latent Dirichlet Allocation applied as an LFM recommender
$SequentialLFM/LDA$	LFM/LDA with sequential context
$TemporalLFM/LDA$	LFM/LDA with temporal context
$TempSeqLFM/LDA$	LFM/LDA with both sequential & temporal contexts

Table 1: **Recommenders' Descriptions**

3 Algorithms

This section describes the recommendation methods that were employed and compared in this work. We start by presenting in Section 3.1 several simple memory-based approaches that are popularity-oriented. Each of these approaches focuses on a different aspect of the data, such as the time-of-day, the current item being watched, and the history of a given device. The objective of these memory-based recommenders is to examine the potential power of various signals by putting them to use with basic and simple approaches. These methods serve as baselines in our evaluation section. We continue by describing in Section 3.2 how two collaborative-filtering methods are employed in order to produce personalized recommendations: one is a *Latent Factor Model* (LFM) and the other harnesses the *Latent Dirichlet Allocation* (LDA) algorithm. Finally, we describe in Section 3.3 how the contexts of the currently watched item and the time of day context are incorporate into these personalized methods. The way we utilize the context of the currently watched item is a main contribution of this work. It explains how we extend the standard factorization inner product into a *3-way* score calculation that exploits the available knowledge of what is being watched on top of a conventionally trained non-contextual model. Table 1 summarizes the recommenders we experimented with including their high level description.

3.1 Memory-Based Popularity Baselines

General Popularity The General Popularity Recommender ($GeneralPop$) is a simple memory based method. This recommender takes the portion of devices that watched an item i to be i's score for all devices, which makes it a non-personalized recommender. The general popularity score for an item i is:

$$GeneralPop(i) = (\#devices\ that\ watched\ i)/\#devices$$

Temporal Popularity A natural refinement to *GeneralPop* is considering the time when the recommendations are to be consumed. As different items are

popular at different times of day, refining the popularity measurement to consider the time of recommendation is expected to improve the relevance of the recommended items. The Temporal Popularity Recommender (*TemporalPop*) measures the item's popularity in a specific time-of-day t, relative to its general popularity. It is a non-personalized recommender that for all devices calculates the score of item i at time t as:

$$TemporalPop(i,t) = Popularity(i|t)/GeneralPop(i),$$

where $Popularity(i|t) = \frac{\#devices\ that\ watched\ i\ @time(t)}{\#active\ devices\ @time(t)}$, and the $time(t)$ is an aggregation of a one-hour-granularity time slot (e.g. 8:00-9:00) over all days in the training data.

Sequential Popularity The main contextual aspect of this work is what to watch next after the current show. The Sequential Popularity Recommender (*SequentialPop*) scores items according to their conditional popularity of being watched sequentially after a specific item. Given the currently watched item c, the score of an item i to be watched next, for any device is:

$$SequentialPop(i|c) = \frac{\#times\ i\ was\ watched\ after\ c}{\#times\ item\ c\ was\ watched}$$

This method is non-personalized as the score reflects the popularity of watching these two items in an ordered sequence independently of a specific device. We also experimented with a personalized version of *SequentialPop*, that only counts the number of times a given device d watched the two items consecutively, but found the training data to be too sparse.

An additional natural baseline is a recommender that simply recommends the next show on the same channel. However, since our data lacks the channel information, we cannot determine whether the user actively changed the channel or stayed on the same channel. The *SequentialPop* recommender is the closest approximation of this baseline, as a high probability of two shows being watched consecutively is a good indication of them airing on the same channel.

Device Popularity The Device Popularity Recommender (*DevicePop*) relies on the assumption that users re-watch items they already watched in the past. As such, it fits well to the *habitual* setting that include recommendations of items previously watched by a given device. *DevicePop* is a personalized memory based method in which the score for device d and item i is:

$$DevicePop(d,i) = \#times\ device\ d\ watched\ item\ i$$

The weak spot of *DevicePop* is on items that were seldom or never watched before on a given device d, as *DevicePop* will induce score ties between large sets of such items. In order to overcome *DevicePop*'s inability to differentiate between such items, we often combined it with an additional recommender by adding their scores together. We denote such combinations as *DevicePop* + X, where X is the combined recommender. Note that *DevicePop* is clearly not suitable for the *exploratory* setting and was not tested in it.

3.2 Collaborative Filtering Methods

As basic personalized recommenders, we use two methods that rely on latent factors or topics: the first is a Latent Factor Model and the second is an application of Latent Dirichlet Allocation. Both methods output two matrices as their resulting models: a $|\mathcal{D}| \times n$ matrix $\mathcal{M}^{\mathcal{D}}$ and an $n \times |\mathcal{I}|$ matrix $\mathcal{M}^{\mathcal{I}}$, where $\mathcal{D}$ and $\mathcal{I}$ are the sets of all devices and items respectively. Each row of $\mathcal{M}^{\mathcal{D}}$ and each column of $\mathcal{M}^{\mathcal{I}}$ are vectors corresponding to a device and an item respectively. We denote the matrix vectors corresponding to device d and item i as $\mathcal{M}^{\mathcal{D}}_d$ (row) and $\mathcal{M}^{\mathcal{I}}_i$ (column) respectively. n is the selected latent dimension that represents n latent factors or topics. In our experiments we found the value n=80 produces the best results. The standard LFM inner product of $\mathcal{M}^{\mathcal{D}}_d$ and $\mathcal{M}^{\mathcal{I}}_i$ reflects the affinity between d and i. Formally, if $\bar{d} = \mathcal{M}^{\mathcal{D}}_d$ and $\bar{i} = \mathcal{M}^{\mathcal{I}}_i$, the recommendation score is calculated as:

$$R(d,i) = \langle \bar{d}, \bar{i} \rangle = \sum\nolimits_{k=1}^{n} \bar{d}_k \cdot \bar{i}_k \tag{1}$$

LFM with Stochastic Gradient Descent As a state-of-the-art recommendation method we used LFM optimized using *Stochastic Gradient Descent*. The cost function we used for optimization is a log-sigmoid function that penalizes watched items with a low score and non-watched items with a high score:

$$cost(d,i) = \begin{cases} -log(Sig(R(d,i))), & \text{if } d \text{ watched } i, \\ -log(1 - Sig(R(d,i))), & \text{otherwise,} \end{cases}$$

where $Sig(x) = \frac{1}{1+e^{-x}}$ and $R(d,i)$ is the score depicted in Equation 1. To avoid overfitting, we used early-stop validation to determine the number of training iterations. This approach is denoted as LFM in the rest of the paper.

LDA as a Collaborative Filtering Recommender To apply LDA on TV data, every device in the data is considered to be an input document for LDA and every item watched by that device is considered to be a word in that document. Multiple watches of the same item by a device correspond to a word appearing multiple times in the document. LDA's assumption that documents belong to multiple topics fits our case well – often multiple users, possibly with varying tastes, share the same TV set. Different combinations of the viewing users may have different "topics" of interest, or "tastes", that describe a given device. LDA models each device as a mixture of topics that relate to the combinations of entities that share the device.

We used an existing LDA implementation[5] to produce the $\mathcal{M}^{\mathcal{I}}$ matrix from which we inferred the $\mathcal{M}^{\mathcal{D}}$ matrix. Each row in $\mathcal{M}^{\mathcal{D}}$ corresponds to the probabilities of a given device to watch each of the n latent topics and each column in $\mathcal{M}^{\mathcal{I}}$ corresponds to the probabilities of a given topic to generate a view of each

[5] LDA package by Daichi Mochihashi, NTT Communication Science Laboratories, *http://chasen.org/%7Edaiti-m/dist/lda/*

of the items in the data. In other words, $\mathcal{M}^{\mathcal{D}}_{d,k} = P(k|d)$ and $\mathcal{M}^{\mathcal{I}}_{i,k} = P(i|k)$. Thus, applying Equation 1 on $\bar{d} = \mathcal{M}^{\mathcal{D}}_{d}$ and $\bar{i} = \mathcal{M}^{\mathcal{I}}_{i}$ provides an estimation of the probability that item i will be watched on device d:

$$R(d,i) = \sum\nolimits_{k=1}^{n} \bar{d}_k \cdot \bar{i}_k = \sum\nolimits_{k=1}^{n} P(i|k) \cdot P(k|d) = P(i|d)$$

3.3 Contextual Personalization

Personalization Using Sequential Context Equation 1 considers only the device d and the item i. Given that we also know the context of a currently watched item c, we can combine it into the recommendation score. Our assumption is that $\mathcal{M}^{\mathcal{I}}_{c}$, the latent representation of c, can provide information regarding the "taste" of the entity currently watching it. Intuitively, promoting agreement with c may refine the reliance on the latent representation of the device, which encapsulates together the "tastes" of multiple viewers. We thus extend the standard inner product calculation into a *3-way* calculation that takes c into account. This contextual recommendation score is the sum of the triple element-wise product of the vectors $\bar{d} = \mathcal{M}^{\mathcal{D}}_{d}$, $\bar{c} = \mathcal{M}^{\mathcal{I}}_{c}$, and $\bar{i} = \mathcal{M}^{\mathcal{I}}_{i}$:

$$R(d,i,c) = \sum\nolimits_{k=1}^{n} \bar{d}_k \cdot \bar{c}_k \cdot \bar{i}_k \tag{2}$$

As *LFM* model values are often negative, we normalized *LFM*'s model $\mathcal{M}^{\mathcal{I}}$ by adding the absolute value of $\mathcal{M}^{\mathcal{I}}$'s minimal entry to all entries, resulting with all non-negative item factors. The $\mathcal{M}^{\mathcal{D}}$ model is unaffected and may contain negative values. This normalization makes sure that for a *negative* user factor $\bar{d}_k$, the final score will decrease as the item factors $\bar{c}_k$ and $\bar{i}_k$ increase and vise versa. Without this normalization, the final score would have decreased when all three factors *agree* with negative values (multiplication of 3 negative values). Note that for using Equation 2 there is no need to change the training procedure or the existing models $\mathcal{M}^{\mathcal{D}}$ and $\mathcal{M}^{\mathcal{I}}$. The modification is only in how the recommendation score is computed. We denote applying the sequential context of the currently watched item on *LFM* and *LDA* as *SequentialLFM* and *SequentialLDA*.

Personalization Using Temporal Context To apply temporal context into our personalized recommenders we take a post-filtering contextual approach. Basically, we combine a given recommender with the temporal context by multiplying the recommendation score $R(d,i)$ with the *TemporalPop* score of item i at time t. Thus, we promote items in times of the day when they are more popular while maintaining the personalized aspect. In case $R(d,i) < 0$ we divide $R(d,i)$ by *TemporalPop* so that a high *TemporalPop* score will improve the recommendation score by making it "less negative". For example, for *LFM*:

(3)

	Total	**Train**	**Test**
Months	4	3	1
Devices	339,647	339,647	311,964
Items	19,546	17,232	11,640

Table 2: **Data set numbers**

$$TemporalLFM(d,i,t) = Temporal(R(d,i),i,t) = \begin{cases} R(d,i) \times TemporalPop(i,t), & \text{if } R(d,i) \geq 0, \\ R(d,i) \div TemporalPop(i,t), & \text{otherwise.} \end{cases}$$

where $R(d,i)$ is the non-contextual LFM score. We denote the LFM and LDA recommenders, combined with a temporal context as *TemporalLFM* and *TemporalLDA*.

Temporal and Sequential Context Composition It is simple to combine the temporal context on top of a sequential context recommender. For example, applying both on the LFM recommender is a composition of the temporal context function over the *SequentialLFM* score:

$$R(d,i,c,t) = Temporal(SequentialLFM(d,i,c),i,t)$$

in accordance with Equations 2 and 3. The composition of temporal context over $SequentialLFM$ and $SequentialLDA$ is denoted as *TempSeqLFM* and *TempSeqLDA*, respectively.

4 Experimental Settings

4.1 The Data

Our analysis is based on broadcast viewership data that is collected from smart TV devices in households within the United States. The raw data is comprised of a set of device id, item id, timestamp triples. The item id uniquely identifies a series, not a specific episode. Neither the identity of the individual watching the show nor the channel on which the show is being broadcast are available to us.

Data was collected for 340,000 devices, which watched 19,500 unique items over a period of 4 months in early 2013 (exact numbers are reported in Table 2). We consider the first 3 months as training data – composed of more than 19 millions device-item pairs, while the last month is used as a test set. In the *habitual* setting, the test set includes all instances of consecutively watched items – a total of 3.8 million item pairs. The *exploratory* setting considers only consecutive items whose latter item was not watched by the device in the training data. There were 1.7 million consecutive item pairs in the test set of this setting.

4.2 Emulating Inventories

A TV recommender system ranks items from an inventory of shows available in a given context. However, the actual inventory available at each household depends on its location (for local channels), provider (cable, satellite), premium channel selections, subscriptions, and more. Our data lacks this information. We therefore need to emulate the unknown inventories.

In the evaluation stage, we go over pairs of items watched on a device in succession in the test set. Given a device d, a pair of consecutive items $\langle c, i\rangle$ watched by d in the test set, and the time t, (when the device finished watching c and started watching i), we emulate $\mathcal{I}_{c,t}$, the inventory of items available for watching after c at time t, as the set of all shows j that were watched by some device while d watched i. In addition, we require that for every show $j \in \mathcal{I}_{c,t}$, the pair $\langle c, j\rangle$ has been watched consecutively by some device in the *entire* data set (train and test portions). This procedure approximates the real TV lineup at time t on devices where show c is available. In the *habitual* setting, this procedure results in inventories comprising of 390 items on average. In addition, in the *exploratory* setting, items watched by the device in the training data are removed from the inventory for this specific device. This reduced the average size of inventories to 345 items.

4.3 Evaluation Metric

The metric used in our empirical evaluation is the *Average Rank Percentile* (ARP) metric. ARP measures how high (on average) the show actually watched next by the device was ranked by the recommender. Formally, given consecutive items $\langle c, i\rangle$ watched at time t by device d, we generate the inventory $\mathcal{I}_{c,t}$ and then rank all items $j \in \mathcal{I}_{c,t}$ using the model scoring function $R(d, j, c, t)$. The *rank percentile* is computed as $(|\mathcal{I}_{c,t}| - r(i) + 1)/|\mathcal{I}_{c,t}|$, where $r(i)$ is i's rank in the output of the model. ARP is the average rank percentile over all pairs of consecutive items.

Since each recommendation instance ranks an inventory containing exactly one "correct" answer (the show actually watched next), the per instance rank percentile is identical to the well-known *Area Under the ROC Curve* (AUC) metric on the ranked inventory. ARP is also somewhat similar to the well-known *Mean Reciprocal Rank* (MRR) metric; however, MRR values across multiple instances are comparable only when the lengths of the ranked lists are fixed. Since inventory sizes vary across different sequential and temporal contexts, the percentile in which the correct item is ranked is a better reflection of recommendation accuracy than its reciprocal rank.

From the ranked inventory list only k items are presented to the users. While in general these are the top-k items, the recommender system might apply a diversification policy or some filtering (like in the case the user asks only for a specific genre) that results in presenting items that are not at the top of the list. Therefore, we prefer the ARP metric for measuring the quality of the entire ranking over other popular metrics such as recall and precision @k that focus on the quality of its top.

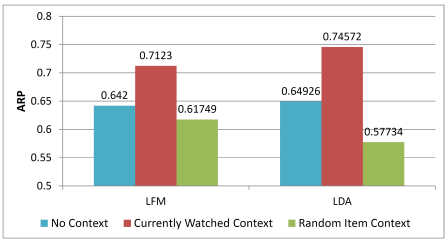


Fig. 1: Sequential context effect (*exploratory* setting)

5 Experimental Results

5.1 Attributing Context to Users

We begin by establishing the usefulness of the currently watched show as a context for recommending what to watch next. To this end, we compared several variations of *LFM* and *LDA* in sequential context (according to Equation 2): (a) using the currently watched item, (b) using a random item previously watched on the device, and (c) using no context. Figure 1 depicts the performance of these recommenders. The improvement in recommendation quality is clear when using the sequential context item, raising the ARP by almost 10% over *LDA*, and by almost 6% over *LFM*. Interestingly, using as a context an item randomly chosen from shows previously watched on the device performs worse than using no context at all. This demonstrates the importance of using the context item from the current viewing session and not an arbitrary item.

To further establish how accurately we attribute the context to a specific user we examine the performance of our recommenders as a function of the number of topics in the device, namely the *device topical entropy*. Intuitively, entropy gives an approximate measure of how diverse are the tastes of the users watching the device. Lacking the information on the number of users per device, we assume a correlation between the taste diversity (number of topics) and the number of users sharing the device.

We use *LDA*'s model $\mathcal{M}^{\mathcal{D}}$ to compute the following entropy measure for a device d:

$$H_d = \sum\nolimits_{0 \le k \le n} -P(k|d) \times \log P(k|d),$$

where $P(k|d) = \frac{\mathcal{M}^{\mathcal{D}}_{d,k}}{\sum_{0 \le k' \le n} \mathcal{M}^{\mathcal{D}}_{d,k'}}$. In other words, $P(k|d)$ is the probability that device d belongs to topic k. A zero entropy value means that the device's users span only a single topic with greater than 0 probability. On the other hand, a uniform distribution of viewed topics on a device maximizes the entropy. For example, when entropy is around 4.5, significant topic probabilities are distributed across more than $2^{4.5} = 22$ different topics. Figure 2 depicts the ARP of devices as a function of their topic distribution entropy. To reduce noise, the results are smoothed – each data point (x, y) represents a "sliding window", corresponding to the average ARP of 200 devices whose median entropy was x.

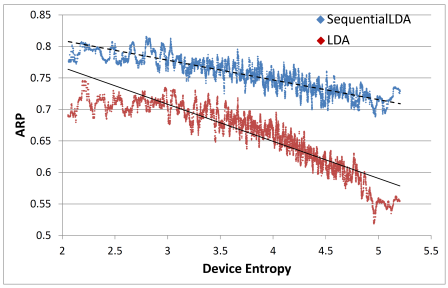


Fig. 2: Profile diversity effect (*exploratory* setting)

ARP results for non-contextual *LDA* demonstrate how hard the recommendation problem is across topical entropy ranges. Devices with low entropy values are more predictable and easier for recommendation generation. As entropy values rise, the taste variance (entropy) increases and the recommendation task becomes more difficult. As can be seen, *SequentialLDA* dominates *LDA* across all entropy ranges. In addition, as device entropy increases, the drop in performance is steeper for the non-contextual recommender. This implies that especially when device predictability is low (high entropy), relying on the currently watched item as context for recommendation increases precision. *SequentialLDA* maintains solid results (ARP > 0.7) even for the highest entropy values, where non-contextual *LDA* degenerates almost to a random recommender (ARP of ~ 0.5). The results are shown only for *LDA*-based recommenders in the exploratory settings, but are consistent with their *LFM* and habitual settings counterparts.

5.2 The Exploratory Setting

In the *exploratory* setting, a recommender should rank items that were never watched on a given device. Figure 3 shows the performance of *LFM* and *LDA* using different combinations of context, in comparison to the baseline recommenders (on the left-hand side).

One can notice that the performance of the *SequentialPop* baseline recommender are not so high. We conclude that a recommender that simply recommends to stay on the same channel will come up with similar results. Our non-contextual *LFM* and *LDA* recommenders are also comparable with *SequentialPop*. Sequential context is shown to improve performance for both *LFM* and *LDA*. Adding temporal context to non-contextual recommenders yields a greater improvement, surpassing the *TemporalPop* baseline. The clear winner in both cases is the recommender that combines both sequential and temporal context: *TempSeqLDA* displays a 27.5% ARP increase over the non-contextual *LDA*.

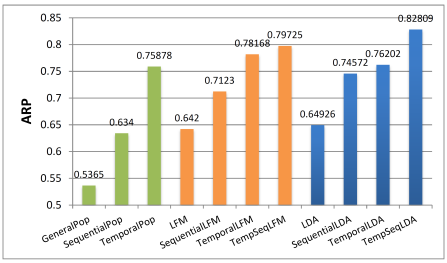
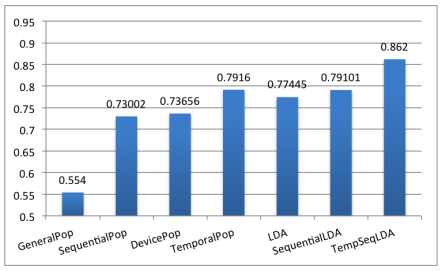
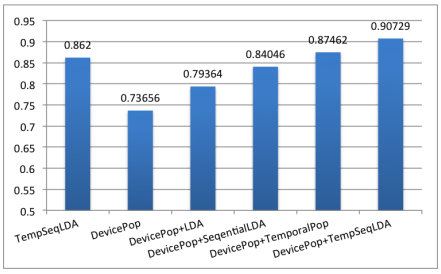


Fig. 3: ARP in the *exploratory* setting



(a) Memory-based and LDA baselines with different contexts



(b) *DevicePop* combined with various recommenders

Fig. 4: ARP in the *habitual* setting

5.3 The Habitual Setting

Figure 4 displays ARP results for the *habitual* setting. Looking at Figure 4a, we find that adding sequential context to the *LDA* recommender gives a minor increase in accuracy, while adding both contexts (*TempSeqLDA*) gives a 11.3% increase over *LDA* and a 9% increase over *SequentialLDA*. In general, the *habitual* setting is considered to be easier as users exhibit repeated consumption patterns of TV shows. This can be observed in the relatively strong performance of non-contextual methods. Still, when considering these methods as baselines, the advantage of combining them with context is evident.

DevicePop is expected to be a solid recommender in the *habitual* setting. However, its main weakness is dealing with previously non-watched items. To counter this weakness, we experimented with combining *DevicePop* with other recommenders. Figure 4b shows results for four such combinations. Combining *DevicePop* with *LDA* and *SequentialLDA* increase ARP by 7.7% and 14% respectively. Combining *DevicePop* with *TemporalPop* yields a 18.7% increase in ARP, surpassing *TempSeqLDA*, which is given for reference on the left-most column. Finally, combining *DevicePop* with *TempSeqLDA* achieves an ARP > 0.9, which demonstrates the power of employing all contexts together.

6 Conclusions and Future Work

This work addresses the recommendation challenge presented by multi-user Smart TV devices through leveraging available context. We show how existing personalization models can tap real time information – temporal and sequential contexts – to significantly improve the recommendation quality of the program to watch next on the device. Specifically, we extend the common matrix factorization inner-product recommendation score into a *3-way* product calculation that considers sequential context – namely, the currently watched item – without changing the standard learning procedure or model.

Our experiments demonstrate that using context significantly improves recommendation accuracy, in both memory-based and collaborative filtering approaches; and that per context used, collaborative filtering schemes outperform the corresponding memory-based counterparts. The positive contribution of context is due to it "narrowing" the topical variety of the program to be watched next on the device. While tapping temporal context alone performs quite well in both the *exploratory* and *habitual* settings, enhancing it with sequential context results in significant additional accuracy gains. Finally, in the *habitual* setting, which is typical for TV, prediction accuracy improves when explicitly accounting for repeated item consumption patterns.

References

1. G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems (TOIS)*, 23, 2005.
2. G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender Systems Handbook*, pages 217–253. 2011.
3. L. Baltrunas and X. Amatriain. Towards time-dependant recommendation based on implicit feedback. In *CARS*, 2009.
4. L. Baltrunas, B. Ludwig, and F. Ricci. Matrix factorization techniques for context aware recommendation. In *RecSys*, pages 301–304, 2011.
5. L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. In *RecSys*, pages 245–248, 2009.
6. R. Bambini, P. Cremonesi, and R. Turrin. A recommender system for an IPTV service provider: a real large-scale production environment. In *Recommender Systems Handbook*, pages 299–331. 2011.
7. P. Bellekens, G.-J. Houben, L. Aroyo, K. Schaap, and A. Kaptein. User model elicitation and enrichment for context-sensitive personalization in a multiplatform tv environment. In *EuroITV*, pages 119–128, 2009.
8. M. Bendersky, L. G. Pueyo, J. J. Harmsen, V. Josifovski, and D. Lepikhin. Up next: retrieval methods for large scale related video suggestion. In *KDD*, pages 1769–1778, 2014.
9. D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.
10. H. Cao, D. Jiang, J. Pei, E. Chen, and H. Li. Towards context-aware search by learning a very large variable length hidden markov model from search logs. In *WWW*, pages 191–200, 2009.

11. C. Chaey. http://www.fastcompany.com/3015138/fast-feed/now-you-can-have-multiple-user-profiles-on-one-netflix-account, 2013.
12. J. Davidson, B. Liebald, J. Liu, P. Nandy, and T. V. Vleet. The youtube video recommendation system. In *RecSys*, pages 293–296, 2010.
13. N. Hariri, B. Mobasher, and R. Burke. Query-driven context aware recommendation. *RecSys*, 2013.
14. B. Hidasi and D. Tikk. Fast ALS-based tensor factorization for context-aware recommendation from implicit feedback. ECML, pages 67–82, 2012.
15. Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272, 2008.
16. A. Karatzoglou. Collaborative temporal order modeling. In *RecSys*, pages 313–316, 2011.
17. A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *RecSys*, pages 79–86, 2010.
18. J. Kleinman. http://www.technobuffalo.com/2013/09/16/android-4-2-multi-user-support-coming-to-some-samsung-tablets/, 2013.
19. Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.
20. S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *SIGIR*, pages 635–644, 2011.
21. A. Said, E. W. D. Luca, and S. Albayrak. Inferring contextual user profiles: Improving recommender performance. In *RecSys Workshop on Context-Aware Recommender Systems*, 2011.
22. Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, and N. Oliver. TFMAP: Optimizing map for top-n context-aware recommendation. In *SIGIR*, pages 155–164, 2012.
23. J. Weston, C. Wang, R. J. Weiss, and A. Berenzweig. Latent collaborative retrieval. In *ICML*, 2012.
24. R. W. White, A. Hassan, A. Singla, and E. Horvitz. From devices to people: attribution of search activity in multi-user settings. In *WWW*, pages 431–442, 2014.
25. B. Xiang, D. Jiang, J. Pei, X. Sun, E. Chen, and H. Li. Context-aware ranking in web search. In *SIGIR*, pages 451–458, 2010.
26. G. Xu, Y. Zhang, and X. Yi. Modelling user behaviour for web recommendation using LDA model. *WI-IAT*, 3, 2008.
27. M. Xu, S. Berkovsky, S. Ardon, S. Triukose, A. Mahanti, and I. Koprinska. Catch-up tv recommendations: show old favourites and find new ones. In *RecSys*, pages 285–294, 2013.
28. A. Zhang, N. Fawaz, S. Ioannidis, and A. Montanari. Guess who rated this movie: Identifying users through subspace clustering. In *UAI*, pages 944–953, 2012.
29. D. Zibriczky, B. Hidasi, Z. Petres, and D. Tikk. Personalized recommendation of linear content on interactive TV platforms: beating the cold start and noisy implicit user feedback. In *Workshop in Conference on User Modeling, Adaptation, and Personalization,*, 2012.
30. J. Zimmerman, K. Kurapati, A. L. Buczak, D. Schaffer, S. Gutta, and J. Martino. Tv personalization system: Design of a tv show recommender engine and interface. In *personalized Digital Television: Targetting Programs to Individual Viewers*, pages 27–51. 2004.