Which App Will You Use Next? Collaborative Filtering with Interactional Context

Nagarajan Natarajan* naga86@cs.utexas.edu Dept. of Computer Science University of Texas at Austin

Donghyuk Shin* dshin@cs.utexas.edu Dept. of Computer Science University of Texas at Austin

Inderjit S. Dhillon inderjit@cs.utexas.edu Dept. of Computer Science University of Texas at Austin

ABSTRACT

The application a smart phone user will launch next intuitively depends on the sequence of apps used *recently*. More generally, when users interact with systems such as shopping websites or online radio, they click on items that are of interest in the *current* context. We call the sequence of clicks made in the current session *interactional context*. It is desirable for a recommender system to use the context set by the user to update recommendations. Most current context-aware recommender systems focus on a relatively less dynamic *representational* context defined by attributes such as season, location and tastes. In this paper, we study the problem of collaborative filtering with interactional context, where the goal is to make personalized and dynamic recommendations to a user engaged in a session. To this end, we propose the iConRank algorithm that works in two stages. First, users are clustered by their transition behavior (one-step Markov transition probabilities between items), and cluster-level Markov models are computed. Then personalized PageRank is computed for a given user on the corresponding cluster Markov graph, with a personalization vector derived from the current context. We give an interpretation of the second stage of the algorithm as adding an appropriate *context bias*, in addition to *click bias* (or rating bias), to a classical neighborhood-based collaborative filtering model, where the neighborhood is determined from a Markov graph. Experimental results on two real-life datasets demonstrate the superior performance of our algorithm, where we achieve at least 20% (up to 37%) improvement over competitive methods in the recall level at top-20.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering*

*Equal contribution to the work.

RecSys'13, October 12–16, 2013, Hong Kong, China.

ACM 978-1-4503-2409-0/13/10 ...$15.00.
http://dx.doi.org/10.1145/2507157.2507186.

Keywords

Collaborative Filtering, Interactional Context, Markov Model, Personalized PageRank, Context-Aware

1. INTRODUCTION

The problem of recommending an item to a user given a sequence of items that the user *recently* interacted with arises in many systems. The problem is best understood by considering a smart phone user navigating through various applications (apps) on the mobile device. Mobile apps are often used in conjunction with other relevant apps. For example, if a user launches the 'contacts' app, the next app is likely to be the 'mail' or the 'messages' app. The current context of the user can be characterized from recently launched apps. Recommending the "right" app to use next based on the context of the user's actions would improve user experience and multitasking efficiency.

Based on the classification presented in [8], we call the context that arises from user's activity within a session *interactional* context. In contrast, most of the existing context-aware recommender systems focus on *representational* context, usually defined *before* the user interacts and provided as attributes such as location, weather and interests [1].

The problem differs from traditional collaborative filtering settings, such as the Netflix rating prediction problem, in many respects. First, user interaction with items such as apps is *brief* and *repetitive* in nature, whereas items like movies are usually watched/rated once. Second, the user feedback is inherently implicit in the form of item clicks, as opposed to explicit feedback like ratings or comments. Additionally, we have a temporal ordering of clicks within user sessions. Third, recommendations must be made available *dynamically* as the user interacts with the system. That is, recommendations should be updated each time a user clicks an item. Finally, we have the notion of interactional context defined by the session in progress, to which the recommendations are targeted. It is desirable for a recommender system to use the context set by the user to update its recommendations. It might be tempting to relate the aspect to that of online collaborative filtering systems [7], where systems could use newly available ratings to recompute predictions. However, there is a subtle difference. The interactional context should help better zero in on the apps that the user would launch next while she interacts with the system. It is therefore necessary to treat the current session differently from past sessions.

The problem is applicable in any setting where items are generally used repeatedly, such as listening to tracks from

online music streaming services or browsing products in shopping websites. If a user has been listening to a series of rock music, she is more likely to prefer other rock music than tracks from a different genre. Note that in such general settings, the system should be able to recommend *new* or *unseen* items to the user. In the next app prediction setting, one could restrict the recommended apps to those that are already installed in the user device, or even recommend new apps that other users have used in similar context.[1] A less obvious but tangible benefit of predicting the next app is that the predicted apps can be pre-loaded to reduce both app launch latency and energy consumption [19].

In this paper, we propose a novel method, iConRank, for collaborative filtering with interactional context. The fundamental observation is that we do not want to treat the sequence of item clicks as raw counts, but as ordered transitions. We model users' transition behavior between items as a Markov chain, where transition probabilities are empirically estimated. A single global Markov model would fall short of capturing diverse transition patterns. On the other hand, a fully personalized Markov model would suffer from extreme sparsity of observed transitions. So, we seek cluster-level Markov models, where the clusters themselves are *behavioral*. That is, we cluster users by their sparse one-step item transition probabilities and compute a representative Markov model per behavioral cluster. We develop our method by introducing a *context bias* to a classical neighborhood-based collaborative filtering model. Our formulation essentially leads to a personalized PageRank [10] on a particular Markov graph, where the so-called "personalization" vector is derived from interactional context.

Our contributions are as follows:

- The problem of incorporating interactional context in collaborative filtering is relatively unexplored (See Section 2). Though the setting is motivated from app launch patterns of smart phone users, it is applicable in many click-based interactive systems.
- We propose iConRank that makes personalized and dynamic recommendations given the current session. Recommendations are updated as the user interacts with the system. We show that the quality of recommendations made by our algorithm is superior to those of competitive methods on two real-life datasets.
- Behavioral clustering of users allows the system to make recommendations using past item transitions of a given user as well as transitions from users with similar navigational patterns.
- Our method is scalable and can handle large problems efficiently. The clustering stage of our algorithm is done offline. Personalized PageRank can be computed in a scalable manner, as detailed in Section 4.3, which enables implementation on devices with limited processing power.

The remainder of the paper is organized as follows. In Section 2, we review some closely related work. We present a formal description of the problem setting in Section 3. In Section 4, we discuss important aspects of the problem that need to be addressed, and develop our algorithm iConRank. Experimental results on two real-life datasets are reported in Section 5. We conclude in Section 6.

[1]This choice is made when the recommender system is deployed. In experiments, we evaluate with the latter option.

2. RELATED WORK

Traditional collaborative filtering techniques use a history of item preferences by a set of users in order to recommend items of interest to a given user. Commonly used classes of methods are neighborhood-based and latent factor models. Neighborhood-based methods involve computing a measure of similarity between pairs of users or items, and obtaining predictions by taking a weighted average of ratings of similar users or items [17, 4]. Another family of model-based collaborative filtering algorithms is based on latent factor models [12, 16], where user preferences are modeled as interactions between unobserved user and item factors.

Previous research has shown that contextual information can enhance the performance of recommender systems in various applications. Based on the classification of context introduced in [8], context can be distinguished into two types: *representational* and *interactional*. The majority of context-aware recommender systems have investigated the use of representational context, such as time, location and weather [1]. Some context-aware neighborhood-based models [5, 13] incorporate such contextual information in the similarity measure between users. Another approach described in [3] introduces item splitting, where item ratings are split into two virtual items based on a given contextual condition. The virtual items are used instead of the original ones in different collaborative filtering algorithms and a rating is predicted for the virtual item corresponding to the current user's context. The context-aware latent factor model in [2] includes attribute-based context variables as biases to appropriately learn the model parameters. In the tensor factorization method proposed in [15], additional dimension for contextual information is added to the standard user-item ratings matrix.

There has been limited work on recommender systems that incorporate interactional context. Hariri, et. al [9] also consider the problem of recommending the next track given a sequence of tracks recently played. Each sequence of tracks in a hand-compiled playlist database is first represented as a sequence of latent topics (using topic modeling). Frequent patterns of topics are discovered from the topic sequences using a pattern-mining algorithm. These sequential patterns are then used to predict relevant topics given a user session. The predicted topics are used to post-filter an initial ranking produced by a traditional recommendation algorithm. The method is applicable only when the number of topics is small or the maximum length of a session is short. Enumerating all possible sequences can be computationally infeasible otherwise. Moreover, recommendations are not personalized as sequential patterns are mined from the entire population. In the domain of mobile application, recent work on predicting app launch [19] uses both representational (location, time) and interactional (app launch) context information to engineer features. However, the approach in [19] uses only the first app in a given session, which is considered as the trigger. In contrast, our approach considers all apps launched in the current session.

3. PROBLEM SETTING

Our problem setting is motivated by smart phone users who exhibit patterns of interaction with the device through mobile apps. The sequence of apps launched by a user defines the *interactional* context of the user's actions. Interactional context arises from the user's activity within a session

and is dynamic. The setting is applicable to any click-based recommender system, where recommendations are updated as the users *click* on an item (Youtube, Spotify, etc).

We refer to the sequence of items accessed by a user over a certain contiguous period as a *session*. In practice, sessions are defined based on the type of activity (e.g. articles read by an online user when she is signed in). Note that we do not consider any temporal aspects of the session, other than the ordering of clicks. Existing context-aware recommender systems focus on attribute-based representational context that is less dynamic and often fixed *before* the start of a session. A given user may have specified a set of preferences globally, but it is often the case that user preferences change between sessions. The goal of this paper is to present recommendations to a given user based on her past sessions, sessions of other users in the system *and* the current session *in progress*. The key aspects of our problem setting are:

1. There are no explicit "likes" or "dislikes" of an item, unlike the case of Netflix ratings. We want to come up with a ranking of items that the user will click next in the current session. Following the recommender systems literature, our setting relies on *implicit feedback* as against the Netflix prize setting that uses *explicit feedback*.
2. Users may be interested in multiple categories of items, but given the sequence of clicks made in the current session, there is an added *context bias* that needs to be accounted for. In general recommender systems, user bias and item bias are accounted for whereas context bias is either ignored or is not applicable.

We would like to emphasize here that though the first aspect in isolation is well-known in the recommender systems community [14, 20], the second aspect has received little attention [9].

3.1 The problem statement

The item recommendation problem in the *Collaborative Filtering with Interactional Context* setting is formally stated as follows. Given a history of sessions $\mathcal{S}$ of users $\mathcal{U} = \{u_1, u_2, \ldots, u_{|\mathcal{U}|}\}$ over a set of items $\mathcal{I} = \{a_1, a_2, \ldots, a_{|\mathcal{I}|}\}$, and a specific user $u \in \mathcal{U}$ with session *in progress* $s = \langle a_{i_1}, a_{i_2}, \ldots, a_{i_t} \rangle$, for some $t \geq 1$, we want to recommend the best candidate item $a_{i_{t+1}} \in \mathcal{I}$. Note that we want the recommendations to be (a) *personalized* to the user u, and (b) relevant to the context of the session s. The classical collaborative filtering systems (with implicit feedback) consider a specific case of the problem where the current session is ignored and the goal is to come up with a set of recommendations based on the click history.

4. PROPOSED METHOD

In our problem setting, we work with implicit feedback in the form of click sequences. A widely-used collaborative filtering approach for implicit feedback data is to simply form a user-item count matrix, where an entry represents the number of times[2] a user has clicked on an item in the past. Any of the collaborative filtering approaches for explicit feedback such as matrix factorization or similarity-based methods can then be applied on the count matrix. However, such a naive approach is not appropriate for reasons manyfold. Most importantly, we do not want to predict any exact rating — we just need a ranking of relevant items. It is inherently hard to gauge user preferences with clicks — lack of an established scale like star ratings makes it tricky to compute similarities between users or items. For a detailed discussion of what prevents a direct use of algorithms designed for explicit feedback, see [14]. We will see later in the experiments (Section 5) that collaborative filtering methods on the count matrix perform poorly.

Some latent factor models for ratings data include biases due to attribute-based context variables such as location to appropriately learn the model parameters [2]. Other than the absence of explicit feedback, such context-aware models pose another immediate challenge — they rely on rating instances for different settings of context variables to learn the appropriate biases. In the case of interactional context, it is not obvious how to succinctly define context variables and obtain associated training examples. Also, it must be efficient to *update* model parameters as the current session progresses in order to provide dynamic recommendations.

In this section, we first describe how we model history in the form of click sequences. Then, we derive our PageRank-based method by incorporating interactional context in an existing collaborative filtering framework. Finally, we present our algorithm iConRank.

4.1 Modeling Implicit Feedback

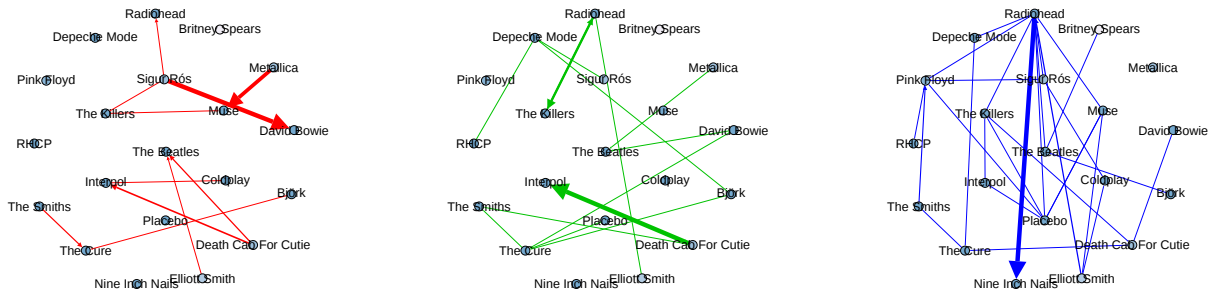
The fundamental observation is that we do not want to treat the session history as counts but as sequences instead. A simple and effective way to model sequences is to use Markov models. Ideally, we would want to know the probability of user clicking on an item given the current session. To this end, we model the users as Markov. The set of items $\mathcal{I}$ corresponds to the state space of the Markov model, and the state transition probability M_{ij} is the probability that item j is clicked immediately after item i. From the sessions data, we can estimate M_{ij} as the fraction of times item j appears immediately after item i, whenever i appears.

Typically, a given user does not have enough training data to estimate $|\mathcal{I}| \times |\mathcal{I}|$ parameters of the Markov model. On the other hand, we do not need to determine a personalized Markov model for each user. The basic idea of collaborative filtering is to combine preferences from "similar" users in order to make recommendations for a given user. One naive way to combine preferences in our setting is to use session data of *all* users to determine a single *global* Markov model. While the training data may be rich enough to estimate a global model, it is less likely to be a good characterization of the diverse transition behavior of users. The aforementioned fully personalized and fully global models fall short, and our approach is to use cluster-level Markov models. It is reasonable to suppose that there are different clusters of users exhibiting a common navigational pattern. To discover behavioral clusters using session data, we need to find a good representation of users, where users who have similar transition patterns are "close" to each other than those that are not.

4.1.1 Behavioral clustering

First, we note that the user-item count matrix itself cannot be used for clustering — we want the clusters to indicate *how* users click and not *what* users click. In particular, we want a feature map Φ_u that encodes the fraction of times a particular transition was made, rather than the number of transitions. Let $\Phi : \mathcal{U} \to \mathcal{M}$, where $\mathcal{M}$ denotes the set of row-stochastic matrices, i.e. $\mathcal{M} = \{M \in \mathbb{R}^{|\mathcal{I}| \times |\mathcal{I}|} : M_{ij} \geq$

[2]Or a monotonic function such as log of the count.



(a) Cluster 1. (b) Cluster 2. (c) Cluster 3.

Figure 1: Behavioral clusters in `lastfm` dataset (see Section 5.1.2). We see how different clusters of users move between the top-20 artists with the highest play counts in the training data. Thicker edges represent higher transition probabilities. Cluster 1 users predominantly move between *Muse, Metallica, David Bowie* and *Sigur Rós*. Cluster 2 users move between *Radiohead* and *The Killers*, and *Interpol* and *Death Cab For Cutie*. Cluster 3 users are marked by transitions to *Nine Inch Nails, Radiohead* and *Placebo*.

$0, \sum_j M_{ij} = 1\}$. In particular, $\Phi(u) = M^{(u)}$ where $M^{(u)}$ denotes the one-step Markov transition probability matrix estimated from the session history of user u. Now, we need a distance measure to cluster users in the space of transition probability matrices. An appropriate measure of distance between two probability distributions is the KL-divergence or the relative entropy. The KL-divergence $d_{KL}(x, y)$ between two p-dimensional probability distributions x and y is defined as:

$$d_{KL}(x, y) = \sum_{i=1}^{p} x_i \log_2 \left(\frac{x_i}{y_i}\right) .$$

The distance between two users u and v is in turn defined as:

$$d(u, v) = \frac{1}{|\mathcal{I}|} \sum_{i=1}^{|\mathcal{I}|} d_{KL}(M_{i\cdot}^{(u)}, M_{i\cdot}^{(v)}) . \quad (1)$$

We have[3] $d(u, v) \geq 0$ since $d_{KL}(.,.) \geq 0$ and $d(u, v) = 0 \iff M^{(u)} = M^{(v)}$. For the actual clustering step, we optimize the k-means objective. The centroid of the cluster π_k is computed as

$$M_k = \frac{1}{|\pi_k|} \sum_{u \in \pi_k} M^{(u)}, \quad (2)$$

where $|\pi_k|$ denotes the number of users assigned to cluster k. Note that $M_k \in \mathcal{M}$ and we use M_k as the Markov model for the kth cluster.

In Figure 1, we show[4] three behavioral clusters discovered in one of our experimental datasets consisting of logs of artists played by users of an online radio station (see Section 5.1.2). The clusters are computed using k-means with the KL-divergence measure (1). Observe that different clusters of users exhibit distinct navigational patterns among the top-20 artists.

4.2 Incorporating interactional context

We motivate our approach from a classical memory-based collaborative filtering model. Memory-based algorithms predict ratings for a given user based on past ratings of the user and other users in the system [17, 4]. The neighborhood models are a classical example, where the predicted rating of an item by a user is given by a weighted combination of "k-nearest neighbor" items (or users). The weights are proportional to the "similarity" between items, which is represented by the vector of observed user ratings. The Pearson correlation coefficient and cosine similarity are two commonly used similarity measures. The similarities between all item pairs are computed offline, and the predicted rating $\hat{r}_{u,i}$ for a user u and an item i is given by

$$\hat{r}_{u,i} = b_{u,i} + \sum_{j \in \mathcal{N}(i)} w(i, j)(r_{u,j} - b_{u,j}) \quad (3)$$

where $\mathcal{N}(i)$ denotes the "neighborhood" set of item i. For example, in the cosine similarity case, the top-k items with highest cosine similarity with item i constitute $\mathcal{N}(i)$. Typically, a rating bias term $b_{u,i}$ is included to account for the user bias (some users are predisposed to rate higher in general) and item bias (some movies get higher ratings than others). The weights are usually normalized for rating prediction tasks.

Let us focus on the item recommendation problem in the interactional context setting, where the knowledge of the current session is available. First, we want to use the cluster Markov models computed using (2) to learn a ranking of items given the user u and the current session s. Second, we want to incorporate interactional context in (3). In particular, we want to introduce a *context bias* term $c_{u,i}$ in addition to the rating bias $b_{u,i}$. In the implicit feedback setting, we interpret $b_{u,i}$ as accounting for *click bias* rather than rating bias. To this end, we want a scoring function $f_{u,i}$ using both past sessions and current session s for the user u. Suitably modifying (3), we have,

$$f_{u,i} = b_{u,i} + \alpha \sum_{j \in \mathcal{N}(i)} w(i, j)(f_{u,j} - b_{u,j}) + (1 - \alpha)c_{u,i} \quad (4)$$

where the nonnegative weight α controls the tradeoff between the current context and information from the past sessions. Notice the recurrence nature of the above equation — we want to estimate all user-item scores *as the session progresses*, as against the context-aware model suggested in [2] based on attribute-based context variables.

Define $z_{u,i} = f_{u,i} - b_{u,i}$ for all $u \in \mathcal{U}$ and $i \in \mathcal{I}$, so that we can remove the click bias from the equation resulting in a simpler model:

$$z_{u,i} = \alpha \sum_{j \in \mathcal{N}(i)} w(i, j) z_{u,j} + (1 - \alpha)c_{u,i}. \quad (5)$$

[3] In practice, many of M_{ij}'s are 0, and to have a well-defined $d_{KL}(.,.)$ we add a relatively tiny value to all the entries of the transition matrices.

[4] For clarity, we only show the top-20 artists with the highest play counts in the training data and omit edges whose transition probability is lower than a certain threshold.

From the graph corresponding to the Markov model with transition probability matrix M, we set $w(i,j) = M_{ji}$. This gives an intuitive interpretation that items that tend to transition to i more often should receive higher similarity than items that do not. Given the choice of $w(i,j)$, $\mathcal{N}(i)$ corresponds to items that are adjacent to i in the Markov graph. Estimating $c_{u,i}$ using the click sequences can be hard and expensive. However, it is easy to specify what items appear in the current session s. We let $c_{u,i} = 1$ if the item a_i appears in s or 0 otherwise. Let $\mathbf{c}_u$ denote the indicator vector with ith entry equal to $c_{u,i}$, and let the vector of item scores for a given user u be $\mathbf{z}_u$. Rewriting (5) in matrix form:

$$\mathbf{z}_u = \alpha M^T \mathbf{z}_u + (1-\alpha)\mathbf{c}_u$$

Letting $\mathbf{1}$ denote vector of all ones, and normalizing $\mathbf{z}_u$ to sum to 1, the above equation can be rewritten as:

$$\mathbf{z}_u = (\alpha M + (1-\alpha)\mathbf{1}\mathbf{c}_u^T)^T \mathbf{z}_u \quad (6)$$

The quantity $\mathbf{z}_u$ is nothing but the personalized PageRank vector for user u, using the graph M and personalization vector $\mathbf{c}_u$. Thus we have an efficient way to estimate $z_{u,i}$. Recall that the click bias $b_{u,i}$ was obviated to compute $\mathbf{c}_u$. So, the final score is given by $f_{u,i} = z_{u,i} + b_{u,i}$. A standard way to estimate click bias is to average the launch count of item a_i over users.[5] However, a better choice would be to use transition probabilities of item a_{i_t} to other items, where i_t is the index of the last item in s, since we focus on item transitions. We find in our experiments that $M^{(u)}_{i_t,i}$ is an effective choice for click bias. The final score for the user-item pair (u,i), given the current session $s = \langle a_{i_1}, a_{i_2}, \ldots, a_{i_t}\rangle$ is:

$$f_{u,i} = z_{u,i} + M^{(u)}_{i_t,i}, \quad (7)$$

where $z_{u,i}$ is the solution to (6).

4.3 iConRank Algorithm

We are now ready to give our algorithm iConRank for recommending items in the collaborative filtering with interactional context setting. Given a current session $s = \langle a_{i_1}, a_{i_2}, \ldots, a_{i_t}\rangle$ executed by a given user u, history of sessions $\mathcal{S}$ for all users in $\mathcal{U}$, and number of behavioral clusters K, the iConRank algorithm is specified as follows:

1. (**Offline step**) Cluster $\mathcal{U}$ using the sessions history $\mathcal{S}$ into K clusters by first forming the per-user transition matrices and then using the k-means algorithm as described in Section 4.1.1. Compute the corresponding Markov transition probabilities M_k for each cluster $k \in [K]$ using (2).
2. Let $\pi(u)$ denote the cluster to which user u belongs. Set $\mathbf{c}_u$ to be the normalized indicator vector of items appearing in s. Compute the personalized PageRank $\mathbf{z}_u$ by (6) with $M = M_{\pi(u)}$ and current $\mathbf{c}_u$.
3. Compute scores $f_{u,i}$, for all $i \neq i_t$ using (7).
4. Rank the items using the computed scores, and return the top-N items as recommendations for $a_{i_{t+1}}$.

A few remarks on iConRank algorithm are in order:

Efficiency: Note that the clustering step is done offline as it is independent of current session s. Steps 2 through 4 are executed *every time* the user clicks, i.e. as session s progresses, so that the recommendations can be updated. Computing the personalized PageRank in Step 2 can be expensive if $|\mathcal{I}|$ is of the order of tens of thousands, even if the matrices M_k are sparse. Scalability is even more of a concern if the algorithm is to be implemented in real-time systems with small processing capabilities like mobile phones. The linearity property of personalized PageRank [10] stated below can be exploited to implement Step 2 in a scalable way. Let $\mathbf{z}_u(\mathbf{v})$ denote the solution to (6) computed with the personalization vector $\mathbf{c}_u = \mathbf{v}$. Then:

$$\mathbf{z}_u(\beta\mathbf{v}_1 + (1-\beta)\mathbf{v}_2) = \beta\mathbf{z}_u(\mathbf{v}_1) + (1-\beta)\mathbf{z}_u(\mathbf{v}_2)$$

for nonnegative β and distributions $\mathbf{v}_1$ and $\mathbf{v}_2$. We can precompute $\mathbf{z}_u(\mathbf{e}_i)$ for $i = 1, 2, \ldots, |\mathcal{I}|$, where $\mathbf{e}_i$ denotes the ith column of the identity matrix. Then, for a given session $s = \langle a_{i_1}, a_{i_2}, \ldots, a_{i_t}\rangle$, and chosen $\mathbf{c}_u = \mathbf{v}$ we can compute $\mathbf{z}_u$ efficiently as:

$$\mathbf{z_u}(\mathbf{v}) = \sum_{j=1}^{t} v_{i_j} \,.\, \mathbf{z_u}(\mathbf{e}_{i_j})$$

It is easy to see why the above equality holds: Our choice of $\mathbf{c}_u = \mathbf{v}$ is such that it is non-zero only in the positions corresponding to items that appear in s, and is normalized to sum to 1.

Cold-start: The algorithm can seamlessly deal with new users. In particular, we can choose $\mathbf{c}_u$ to be the uniform distribution over items and use the global Markov model instead of $M_{\pi(u)}$.

5. EXPERIMENTAL EVALUATION

In this section we present experimental results of our proposed algorithm, iConRank, on two real-life datasets: `Apps` and `LastFM`. We first give a detailed description of the two datasets and the experimental setting. Next we compare the recommendation performance of iConRank to a number of other successful methods. We also evaluate the methods on recommending existing and new items and study how performance is influenced by current session length.

For testing, we measure the accuracy of recommending the next item $a_{i_{t+1}}$, given the current session $s = \langle a_{i_1}, a_{i_2}, \ldots, a_{i_t}\rangle$ of a user. Thus, there is only one correct item for each test case. A user is usually presented with a small list of recommended items. Therefore, we measure *recall at top-N* (Recall@N) for $N = 5, 10, 15, 20$. For the sake of completeness, we also report the *ROC curve* (recall vs. precision plot up to $N = 1,000$), which is the standard measure for comparing the recommendation qualities of different methods for recommender systems. Recall and precision are measured in the standard way for top-N recommendation tasks [6].

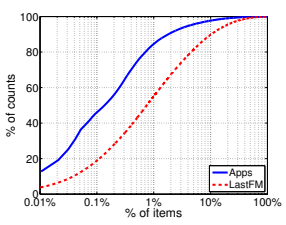
5.1 Dataset Description

The datasets used in this study consist of user event logs with timestamps, where the events are launching mobile apps or streaming tracks of an artist. Sessions are formed from the user logs. We note that consecutive uses of the same item are considered as a single interaction for methods using sequences, including ours. However, launch counts are retained for use by methods based on the user-item matrix.

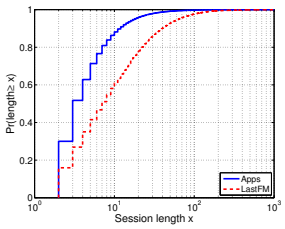
Statistic	`Apps`	`LastFM`
# of users	17,062	941
# of items (apps/artists)	9,583	98,412
# of training sessions	1,167,171	644,001
# of testing sessions	459,899	95,038
Average session length	6.53	18.33
Median session length	3	7

Table 1: Statistics of `Apps` and `LastFM` datasets.

[5] Note that user bias need not be added, as we only need to rank items.



(a) Item count distribution (most frequent item on the left).



(b) CDF of the length of sessions.

Figure 2: Item count and session length distributions for `LastFM` and `Apps` datasets.

Training data consists of all sessions before a chosen date.[6] Table 1 shows relevant statistics of each dataset.

5.1.1 Apps Dataset

The `Apps` dataset is a proprietary dataset obtained from a manufacturer of mobile devices. It consists of mobile app usage patterns of smart phone users.[7] The dataset spans over a one year period and each log record consists of user, timestamp information and one of the two events, app launch and screen on/off. The screen on/off event is used as an indicator of a new session, i.e., a new session is started when the screen is on and the current session is ended when the screen is off *and* at least one minute has elapsed.[8] Figure 2(a) plots the distribution of item count, which corresponds to launch count of apps. Apps in the horizontal axis are ordered by their launch counts, with the most frequently launched app on the left. We can see in Figure 2(a) that top 1% of the most frequently launched apps cover about 85% of total launch counts. Figure 2(b) shows the CDF of the length of sessions. About 87% of sessions have at most length 10. Our statistics align with the findings in [18] that interactions with smart phones are mostly brief.

5.1.2 LastFM Dataset

The `LastFM` (`last.fm`) dataset consists of listening habits of about 1,000 users, where each log contains user, track, artist and timestamp. It has been used in many studies [11, 20]. Here, the task is to recommend the next artist to listen to. Unlike the `Apps` dataset, the `LastFM` dataset lacks any explicit indication of start or end of a session. So, we mark sessions by periods of inactivity. We end a session if there is no other artist streamed within an hour from the last artist and start a new one with the next artist.[8] Listening to music is a more time consuming activity than using a smart phone. Furthermore, preference over artists is more diverse than the apps case, where pre-installed apps are more commonly used. These differences show up in Figure 2. Sessions in `LastFM` tend to be longer and more diverse — about 50% of sessions have at least length 10 and top 1% of the most frequent artists cover only about 55% of total counts.

5.2 Results

We first analyze the behavioral clusters obtained. Then we compare the performance of iConRank to baseline and other context-aware collaborative filtering methods.

[6]Date was chosen to maximize the number of users appearing in both training and testing sets.

[7]We emphasize that the data provided to us was highly anonymized and contained only generic identifiers that cannot be correlated or traced back to actual users.

[8]Time lapses were tuned to achieve the most meaningful session statistics.

Dataset	Cluster	Users	nnz(M_k)	Session Length	
				Avg.	Median
`Apps`	1	4,695	68,967	6.37	3
	2	5,711	73,121	6.32	3
	3	6,656	78,963	6.83	3
`LastFM`	1	327	1,733,056	19.15	7
	2	320	1,500,913	18.55	7
	3	294	1,227,933	17.01	7

Table 2: Statistics of clusters in `Apps` and `LastFM` datasets.

Cluster1	Cluster2	Cluster3
(contacts,phone)	(phone,dialer)	(phone,dialer)
(message,contacts)	(*search,browser*)	(1,phone)
(contacts,dialer)	(*mail,browser*)	(phone,2)
(*settings,phone*)	(message,mail)	(phone,3)
(*data,settings*)	(*calendar,mail*)	(4,phone)
(*camera,photo*)	(*browser,video*)	(5,phone)
(*contacts,settings*)	(*phone,browser*)	(message,mail)

Table 3: Most frequent app transitions of each cluster. (Some apps in the dataset do not have names, so we refer to them by numbers.)

5.2.1 Discovered clusters

We cluster the users as described in Section 4.1.1. In our experiments, we set the number of clusters $k = 3$. We note that there was no significant difference in terms of performance with larger k. To help characterize each cluster, we summarize some statistics of each cluster in Table 2. For the `Apps` dataset, the first cluster is smaller than the other two, but has longer sessions than the second cluster. The third cluster has the largest nnz(M_k), i.e., number of nonzeros in M_k or the number of unique transitions. For the `LastFM` dataset, the clusters are of similar sizes, but the number of unique transitions is relatively distinct.

For the `Apps` dataset, we report the most frequent transitions of each cluster in Table 3. As expected, transitions occur between related apps, such as contacts and phones, browser and search, etc., and it is apparent from Table 3 that each cluster represents a distinct type of app transition behavior. For example, users in the first cluster can be viewed as power users who transition between apps like 'settings', 'data' and 'camera'. In contrast, the second cluster has shorter sessions as reported in Table 2, and transitions are concentrated on basic apps such as 'browser', 'search' and 'calendar'. For the `LastFM` dataset, we visualize behavioral patterns for each cluster in Figure 1 (see Section 4.1.1).

5.2.2 Performance Comparison

We compare iConRank to the following collaborative filtering methods:

- NNCosNgbr: Non-Normalized Cosine Neighborhood [6] is a neighborhood-based model using cosine similarity between items (represented as vectors of launch counts). The score for item a_i for a given user is calculated as a weighted average of the k-nearest neighbor items of a_i as in (3), where we set $k = 5$ and use cosine similarity as the weights.
- SVD(r): The scores are computed as $\hat{R} = U\Sigma V^T \approx A$, where $A \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$ is the user-item count matrix, r is the number of latent factors, $U \in \mathbb{R}^{|\mathcal{U}|\times r}$, $V \in \mathbb{R}^{|\mathcal{I}|\times r}$ are orthonormal matrices and $\Sigma \in \mathbb{R}^{r\times r}$ is a diagonal matrix of the top r singular values. Missing values of A are typically set to zero. The method has been shown

(a) Apps dataset.

Method	Recall@N			
	$N=5$	$N=10$	$N=15$	$N=20$
NNCosNgbr	0.4301	0.5478	0.6167	0.6636
SVD	0.4574	0.5853	0.6480	0.6851
Markov	0.4592	0.5744	0.6370	0.6754
ContextNgbr	0.5266	0.6248	0.6739	0.7045
SeqPattern	0.5517	0.6451	0.6899	0.7223
iConRank	**0.6701**	**0.7927**	**0.8386**	**0.8632**

(b) LastFM dataset.

Method	Recall@N			
	$N=5$	$N=10$	$N=15$	$N=20$
NNCosNgbr	0.0691	0.1044	0.1328	0.1560
SVD	0.0810	0.1286	0.1633	0.1922
Markov	0.0631	0.0905	0.1113	0.1285
ContextNgbr	0.0597	0.0775	0.0884	0.0971
SeqPattern	0.0371	0.0536	0.0656	0.0748
iConRank	**0.1277**	**0.1882**	**0.2304**	**0.2633**

Table 4: Recall@N results on Apps and LastFM datasets for $N = 5, 10, 15, 20$. iConRank outperforms the other methods in all cases, achieving 20% and 37% improvement in Recall@20 over the next best method on Apps and LastFM datasets, respectively.

to achieve good performance in top-N recommendation tasks [6].

- Markov: The recommended item a_{t^*} depends on the last item a_t in the current session via the global Markov model. Formally, $t^* = \text{argmax}_j M_{tj}$, where M is the global transition probability matrix estimated from the sessions of all users.
- ContextNgbr: Identical to NNCosNgbr except that the k-nearest neighbors are computed from the set of items in the current session. We use $k = 5$ (include all items in the current session if its length is less than k).
- SeqPattern: Sequence mining algorithm used in [9].[9] We use the last 10 items as the user's active session.

The last three methods are context-aware approaches that consider user's current session. We note that ranking items by popularity (i.e. number of times an item is accessed in the training phase), which is known to perform reasonably well in recommendation tasks, does significantly worse in our setting than the methods we compare here.

Table 4 reports the Recall@N of the methods on both datasets for $N = 5, 10, 15, 20$. Our algorithm, iConRank, significantly outperforms all other methods with impressive recall rates. It achieves the highest recall of 0.8632 for the Apps dataset, which means the app $a_{i_{t+1}}$ that a user will use has a probability of about 86% to be ranked in the top-20 results returned by iConRank. The next best performing method for the Apps dataset is SeqPattern followed by ContextNgbr, both of which use current context. However, they are the two worst performing methods for the LastFM dataset. This can be expected as the number of items is much larger than in the Apps dataset. SeqPattern suffers from the sheer diversity of artists and lack of adequate representative patterns in the training data. Among the two collaborative filtering based techniques, SVD performs better than NNCosNgbr in both the datasets confirming the findings of [6].[10] Surprisingly, the simple Markov model performs better than collaborative filtering based methods for the Apps dataset, but fails in the LastFM dataset conforming to our intuition that a single global Markov model is not enough. In evaluating the methods on the Apps dataset, the recommended candidate apps for a given user can contain apps that are *not* installed in the user's device, which is consistent with the evaluation on the LastFM dataset.[11]

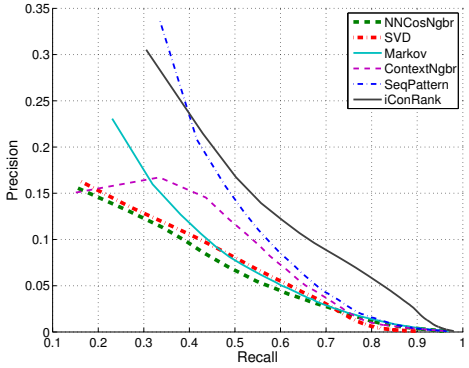
[9] [9] applies sequence mining on latent topics discovered from playlists, but we mine sequence of items directly as we don't have item features.

[10] We tested SVD with various ranks r, and chose $r = 200$ for the Apps dataset and $r = 300$ for the LastFM dataset. Higher ranks yielded only marginal performance improvements.

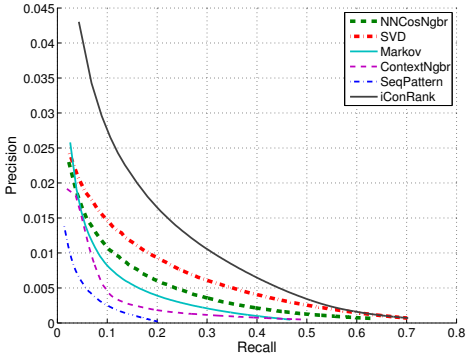
[11] The Apps dataset did not contain information about which apps are installed in the user device.

The ROC curve (up to top-1,000 recommendations) is given in Figure 3. iConRank generally achieves the highest precision for any given recall. One exception is SeqPattern which achieves slightly better recall (about 0.05 higher) at top-1 for the Apps dataset. However, it immediately starts to degrade in performance falling below iConRank for $N > 1$. We further study the performance in different categories: existing and new items. Specifically, evaluation is restricted to items that appeared at least once in the training data ("existing") or those that were not seen in the training data ("new") for each user. The results are presented in Figure 4. We can see that iConRank not only performs well on existing items, but is also capable of recommending new items to users significantly better than the rest.

A desirable property of our algorithm is that it is able to make more accurate recommendations as more items are observed in a given session. We examine how iConRank per-



(a) Apps dataset.



(b) LastFM dataset.

Figure 3: ROC curve for Apps and LastFM datasets up to top-1,000 recommendations. iConRank achieves highest precision-recall rates.

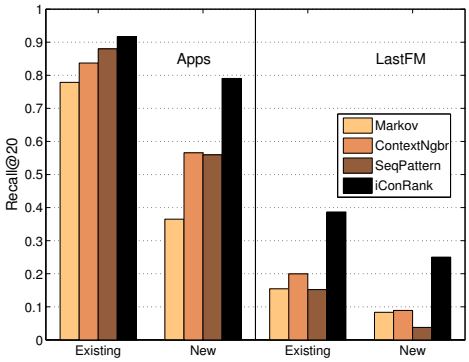


Figure 4: Recall@20 for existing and new items. iConRank performs the best in all cases.

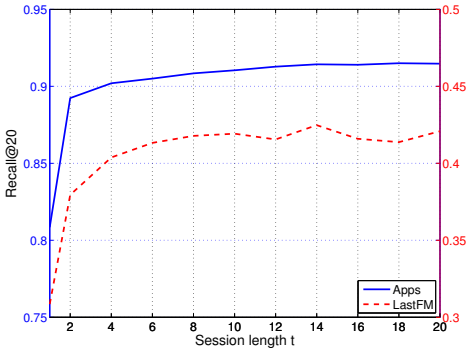


Figure 5: Recall@20 of different session lengths. Performance increases as the session length increases showing that iConRank effectively captures the current context.

forms as more of the current session is revealed. In particular, we measure Recall@20 when t (length of s) varies as shown in Figure 5. As the session length increases, Recall@20 also increases for both datasets, illustrating that iConRank effectively captures the current context of the session. Especially, the performance greatly increases after observing just two items in a given session. However, we also find that recall rates can drop when t becomes very large (not shown in the figure) — the personalization vector tends to a uniform distribution and context ceases to have relevance. In practice, most of the sessions are short as shown in Figure 2(b).

6. CONCLUSIONS

In this paper we have considered the problem of collaborative filtering with interactional context to recommend potential items of interest to a user already engaged in a session, using past sessions of the user and of other users. We are given implicit feedback in the form of clicks and a session that establishes the current context of a user. Our problem setting differs from the traditional setting of collaborative filtering in crucial respects. We propose iConRank, a novel method that is motivated by introducing 'context bias' in the neighborhood-based model. Our formulation essentially leads to the personalized PageRank, where context is captured by the personalization vector. Experimental results on real-life datasets demonstrate that our algorithm achieves superior recommendation performance illustrating its ability to capture the context of a given session.

As part of future work, we plan to study how to combine both interactional context and representational context such as location and time. Exploring other collaborative and content filtering methods to incorporate interactional context are promising research directions.

7. ACKNOWLEDGMENTS

This research was supported by NSF grant CCF-1117055.

8. REFERENCES

[1] L. Baltrunas. *Context-aware collaborative filtering recommender systems*. PhD thesis, Free University of Bozen-Bolzano, 2011.

[2] L. Baltrunas, B. Ludwig, and F. Ricci. Matrix factorization techniques for context aware recommendation. In *ACM RecSys*, pages 301–304, 2011.

[3] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. In *ACM RecSys*, pages 245–248, 2009.

[4] R. M. Bell and Y. Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *ICDM*, pages 43–52, 2007.

[5] A. Chen. Context-aware collaborative filtering system: predicting the user's preferences in ubiquitous computing. In *CHI EA*, pages 1110–1111, 2005.

[6] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-N recommendation tasks. In *ACM RecSys*, pages 39–46, 2010.

[7] E. Diaz-Aviles, L. Drumond, L. Schmidt-Thieme, and W. Nejdl. Real-time top-N recommendation in social streams. In *ACM RecSys*, pages 59–66, 2012.

[8] P. Dourish. What we talk about when we talk about context. *Personal and Ubiquitous Computing*, 8(1):19–30, 2004.

[9] N. Hariri, B. Mobasher, and R. Burke. Context-aware music recommendation based on latent topic sequential patterns. In *ACM RecSys*, pages 131–138, 2012.

[10] T. H. Haveliwala. Topic-sensitive PageRank: a context-sensitive ranking algorithm for web search. *IEEE Trans. on Knowl. and Data Eng.*, 15(4):784–796, 2003.

[11] B. Hidasi and D. Tikk. Fast ALS-based tensor factorization for context-aware recommendation from implicit feedback. In *ECML PKDD*, pages 67–82, 2012.

[12] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *IJCAI*, pages 688–693, 1999.

[13] R. Hu, W. Dou, and J. Liu. A context-aware collaborative filtering approach for service recommendation. In *CSC*, pages 148–155, 2012.

[14] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272, 2008.

[15] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. In *ACM RecSys*, pages 79–86, 2010.

[16] Y. Koren, R. M. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[17] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295, 2001.

[18] C. Shepard, A. Rahmati, C. Tossell, L. Zhong, and P. Kortum. LiveLab: measuring wireless networks and smartphone users in the field. *SIGMETRICS Performance Evaluation Review*, 38(3):15–20, 2011.

[19] T. Yan, D. Chu, D. Ganesan, A. Kansal, and J. Liu. Fast app launching for mobile devices using predictive user context. In *MobiSys*, pages 113–126, 2012.

[20] D. Yang, T. Chen, W. Zhang, Q. Lu, and Y. Yu. Local implicit feedback mining for music recommendation. In *ACM RecSys*, pages 91–98, 2012.