Context Adaptation in Interactive Recommender Systems

Negar Hariri
DePaul University
Chicago, IL 60604, USA
nhariri@cs.depaul.edu

Bamshad Mobasher
DePaul University
Chicago, IL 60604, USA
mobasher@cs.depaul.edu

Robin Burke
DePaul University
Chicago, IL 60604, USA
rburke@cs.depaul.edu

1. ABSTRACT

Contextual factors can greatly influence the utility of recommendations for users. In many recommendation and personalization applications, particularly in domains where user context changes dynamically, it is difficult to represent and model contextual factors directly, but it is often possible to observe their impact on user preferences during the course of users' interactions with the system. In this paper, we introduce an interactive recommender system that can detect and adapt to changes in context based on the user's ongoing behavior. The system, then, dynamically tailors its recommendations to match the user's most recent preferences. We formulate this problem as a multi-armed bandit problem and use Thompson sampling heuristic to learn a model for the user. Following the Thompson sampling approach, the user model is updated after each interaction as the system observes the corresponding *rewards* for the recommendations provided during that interaction. To generate contextual recommendations, the user's preference model is monitored for changes at each step of interaction with the user and is updated incrementally. We will introduce a mechanism for detecting significant changes in the user's preferences and will describe how it can be used to improve the performance of the recommender system.

Keywords

Context-aware recommendation; Collaborative filtering; Multi-armed bandit problem; Thompson sampling

2. INTRODUCTION

Context-aware recommender systems have been the focus of various research studies in the past few years. The main motivation for designing these systems is that the contextual factors can influence users' preferences. Therefore, the *utility* or usefulness of each recommended item does not remain static and changes over time based the current context of the user.

RecSys'14, October 6–10, 2014, Foster City, Silicon Valley, CA, USA.

http://dx.doi.org/10.1145/2645710.2645753 .

The system may not always have complete knowledge of the contextual factors and their values at all times. Based on the classification presented in [1], knowledge of a recommender system about context can be of three types: fully observable, partially observable, and unobservable. The contextual information is fully observable if all the relevant contextual factors, their structure and values are explicitly known. On the other hand, if only part of this knowledge is available, then it is partially observable to the system. For example, if a restaurant recommender system knows that location, and time are the only important contextual factors and the values of these factors are explicitly given to the system, the contextual knowledge is fully observable in the system. But often part of this information might be missing or unknown at design time making context only partially observable. In the case of unobservable knowledge, no explicit information is available about the contextual factors.

If the system has full knowledge of the contextual factors, a representational approach to modeling context [6] can be used where context is defined and represented as a specific set of attributes of the environment within which the user's interaction with the system takes place. On the other hand, if context is unobservable or only partially observable, a recommender designed based on the representational view will not be able to adapt to unforeseen contextual situations that arise dynamically. However, the activity that arises from the context can be observed in terms of changes in user preferences and behavior. Therefore, in such situations, an interactional view to modeling context [6] may be the more appropriate.

Although representational context has been extensively investigated [11][3][9], there has been less focus on interactional systems in context-aware recommendation research. This is partly due to the fact that it is more difficult to capture the contextual information when it is not known to the system at design time.

In this paper, we investigate this problem using an interactive recommender system framework that can dynamically adapt to changes in context. At each step of the interaction, the system presents a list of recommendations and the user provides feedbacks for the recommended items. The feedbacks can be explicit, such as item ratings, or it can be implicitly inferred from user's actions such as a click-through, lingering on a Web page, or purchasing a product. The user's feedback is an indication of the recommendation's utility and is used by the recommender to update the user's preference model after each interaction.

The main objective of the proposed recommendation approach is to identify the recommendation list at each step such that the average obtained utility is maximized over the interaction session with the user. Choosing the best recommendation list at each step requires a strategy in which the best trade-off between *exploitation* and *exploration* is achieved. During an exploitation phase, the recommender system generates the recommendation list maximizes the immediate utility for the current step. During exploration, the system may recommend items that do not optimize the immediate utility, but reduce the uncertainty in modeling a user's preferences and eventually help maximize the utility over the whole session.

This problem can be formulated as a *multi-armed bandit problem (MAB)* which has been studied in various systems and applications [5][10]. However, considering users' changes in context when designing bandit strategies has received very little attention despite the fact that changes in users' preferences can greatly impact the performance of interactive recommender systems. For example, consider an online radio application that receives users' feedback for each song played in a playlist. Assume that a user starts the session while reading but later changes her context to exercising. This change in context can greatly affect the type of feedback the user provides for the recommendations. If the recommender does not distinguish the feedbacks before and after the context transition, it may take many interactional steps to adapt the recommendations to the new contextual state of the user.

In this paper, we introduce an interactive context-aware recommendation algorithm that takes into account the user's interactional context. Our algorithm tracks changes in a user's feedback behavior and detects any sudden significant changes in the user's preferences. It will then adapt the recommendations to match the user's most current preferences. In our experiments, we evaluate the precision of our method in detecting contextual changes and investigate the extent to which our context-aware recommendation approach can provide improvements over non-contextual baselines.

3. RELATED WORK

Several researchers have previously investigated the use of contextual information in various recommendation applications [13, 1]. Many of these studies assume a representational view to modeling context [11, 3, 9]. In [11], a context-aware playlist recommendation system is proposed which provides users the opportunity to browse their music by mood. The system described in [3], focuses on social context and proposes Poolcasting as a method to customize musical sequences for groups of listeners. The music recommender introduced in [9], recommends music suited for places of interest. The contextual recommender described in [8] assumes that context is explicitly given as a query to the system. Our approach is different from [11], [3], [9], and [8] as the contextual factors are not pre-specified in our system.

In [7], a context-aware music recommender is introduced which infers contextual information based on the most recent sequence of songs liked by the user and their associated top frequent tags mined from social tagging Web sites. The system introduced in this paper is different from [7] in various aspects: the recommender presented in this paper does not rely on the item meta-data such as tags (which may not always be available). Instead, contextual changes are inferred just based on users' feedbacks on items. Also, the presented recommender in this paper optimizes the utility over the whole user session rather than individual interactions.

Bandit algorithms have been widely used for the application of producing personalized recommendations [5, 10]. The news recommender introduced in [10] adapts to the temporal changes of existing content. In the proposed system, both users and items are represented as sets of features. Features of a user can include demographic attributes as well as features extracted from the user's historical activities such as clicking or viewing items while item features may contain descriptive information and categories. They formulate this problem as a bandit problem with linear payoff and introduced an upper confidence bound bandit strategy which is shared among all the users. Similarly, the system in [5] uses a bandit algorithm based on Thompson sampling for the task of news recommendation and display advertisement. Similar to [10], several features are used to represent the user and the items and the recommender learns a single bandit strategy for all users while representing each user as a fixed set of features. Our approach is different from [10] and [5] as we are learning a bandit strategy for each user. While these two methods focus on adapting to the changes in content (or items), we are interested in capturing and adapting to the changes in users' interests. Representing each user as a fixed set of features, as in [10] and [5] would prevent us from detecting changes in the users' preferences and capturing interactional context. However, our approach enables us to monitor changes in the users' behaviors.

[4] uses a multi-armed bandit approach for mobile contextual recommendation.The main difference of our approach with [4] is that we are not taking a representational view towards context, instead we are interested in capturing change of behavior which can be induced by contextual changes.

4. INTERACTIVE CONTEXT-AWARE RECOMMENDATION

Interactive recommenders have been extensively used in a variety of application domains. These systems usually rely on an online learning algorithm that gradually learns users' preferences. At each step of the interaction, the system generates a list of recommendations and observes the user's feedback on the recommended items indicating the utility of the recommendations. The goal of such a system is to maximize the total utility obtained over the whole interaction session.

4.1 Multi-Arm Bandit Approach to Online Recommendation

Choosing the best strategy to select the recommendation list at each step can be formulated as a *multi-armed bandit problem (MAB)*. Consider a gambler who can choose from a row of slot machines known as one-armed bandits. The gambler receives a reward from playing each of these arms. To maximize his or her benefit, the gambler should decide on which slot machines to play, the order of playing them and the number of times that each slot machine is played. A similar type of decision problem occurs in many real-world applications including interactive recommender systems. For an interactive recommender, the set of arms corresponds to a collection of items that can be recommended at each step,

and the rewards correspond to the user's feedback, such as ratings or a clickthrough on a recommended item.

An important consideration in a bandit algorithm is the trade-off between *exploration* and *exploitation*. Exploitation is referred to the phase where the available information gathered during the previous steps is used to choose the most profitable item. During exploration, the most profitable alternative may not necessarily be picked but the algorithm may choose other items in order to acquire more information about the preferences of the user which would help to gain more rewards in future interactions.

Several techniques have been proposed for solving the multi-armed bandit problem. Optimal solutions for multi-armed bandit problems are generally difficult to obtain computationally. However, various heuristics and techniques have been proposed to compute sub-optimal solutions effectively. The ϵ-*greedy* approach is one of the most famous and widely used techniques for solving the bandit problems. At each round t, the algorithm selects the arm with the highest expected reward with probability $1-\epsilon$, and selects a random arm with probability ϵ. The *Upper Confidence Bounds (UCB)* class of algorithms, on the other hand, attempt to guarantee an upper bound on the total regret (difference of the total reward from that of an optimal strategy).

Thompson Sampling is another heuristic that plays each arm in proportion to its probability of being optimal. As we are using this heuristic in our system, we briefly describe it in more details. In the next section, we will discuss how this heuristic is used in our system to generate the recommendation list at each interaction step.

The reward (or utility) distribution for each item can depend on various factors including characteristics of the item, general preferences of the user, and the current context of the user. Let $p(r|a,\theta)$ represent the utility distribution for item a, and θ indicate the unknown parameter characterizing the distribution. Also, let $E_r(r|a,\theta)$ represent the expected reward for the item a for the given θ.

At the first step of an interaction, the set of observations, denoted by D, is empty and $p(\theta|D)$ is initialized with a prior distribution for θ. As more rewards are observed at each step, Bayesian updating is performed to update the θ distribution based on the observed rewards.

Algorithm 1 describes the Thompson sampling procedure [5, 15]. At each step, θ is drawn as a sample from $p(\theta|D)$. The expected reward for each of the items is then computed and the arm having maximum expected reward is played and the reward for that arm is observed. The selected arm and the observed reward is then added to the observations set.

Algorithm 1 Thompson Sampling

```
D=∅
for t = 1 to T do
    Draw θ^t ~ P(θ|D)
    Select a_t = argmax E_r(r|a, θ^t)
    Observe reward r_t
    D = D ∪(a_t, r_t)
end for
```

4.2 Context-Aware Recommendation Using Thompson Sampling

We adapt Thompson sampling heuristic as the bandit strategy for generating a recommendation list at each step of interaction with a user. In this setting, θ which characterizes the utility distribution for each item, represents a user's preference model. It is a k-dimensional random vector drawn from an unknown multivariate distribution. The user model is updated after each interaction.

Linearly parameterized bandits[14] are a special type of bandits where the expected reward of each item is a linear function of θ. It has been shown that Thompson sampling with linear rewards can achieve theoretical regret bounds that are close to the best lower bounds [2]. In our approach we also assume that the rewards follow a linear Gaussian distribution.

Given d observations, let $r \in R^d$ be the observed rewards for the recommended items and let F represent a $d \times k$ constant matrix containing the *features* of the recommended items. For example, given two observations (a_1, r_1) and (a_2, r_2), $r = [r_1, r_2]$ has two elements r_1 and r_2 and $F = [f_{a_1}, f_{a_2}]$ where f_{a_i} represents the k-dimensional feature vector for item a_i. The item features can be obtained in various ways. They can be extracted based on the available content data for the items or can be extracted from the users' feedback using different techniques such as matrix factorization. Given a training data set represented as a matrix of users' preferences on items, we apply Principal Component Analysis to represent each item in a k-dimensional space (where $k \ll$ the number of users in the training data).

Assuming a linear Gaussian distribution, we have the following prior and likelihood distributions:

$$p(\theta) = \mathcal{N}(\theta; \mu_\theta, \Sigma_\theta) \tag{1}$$

$$p(r|\theta) = \mathcal{N}(r; F\theta, \Sigma_r) \tag{2}$$

Also, we assume Σ_θ and Σ_r are diagonal matrices with diagonal values of σ_θ, and σ_r respectively.

Given a linear Gaussian system, the posterior $p(\theta|r)$ can be computed as follows (see [12] for a proof):

$$p(\theta|r) = \mathcal{N}(\mu_{\theta|r}, \Sigma_{\theta|r}) \tag{3}$$

$$\Sigma_{\theta|r}^{-1} = \Sigma_\theta^{-1} + F^T \Sigma_r^{-1} F \tag{4}$$

$$\mu_{\theta|r} = \Sigma_{\theta|r}[F^T \Sigma_r^{-1}(r) + \Sigma_\theta^{-1}\mu_\theta] \tag{5}$$

At each step t, θ^t is drawn as a sample from the posterior distribution. The expected reward for each item x is then estimated as $r_x = f_x \cdot \theta$, where f_x represents the extracted features for the item x. The items are then ranked based on the estimated expected reward values. The item with the highest reward is recommended to the user.

4.3 Adaptation to Contextual Changes

This section describes our approach for adapting the recommendations to contextual changes. Our method has two main steps. The first step, is to detect context transitions and the second step is to adapt the recommendations based on the detected transitions.

Our system includes a change detection module that tracks users' interactions and detects any sudden and significant changes in the users' behaviors. At each step t of interaction

with a given user u, the change detection module models the user's interactions in interval $I_t = (t - N, t]$ as well as the interactions in the interval $I_{(t-N)} = (t-2N, t-N]$, where N is a fixed pre-specified parameter representing the length of the interval. These two windows are modeled by computing the following posterior distributions:

$$W_{I_t} = p(\theta|I_t) = \mathcal{N}(\mu_t, \Sigma_t) \quad (6)$$

$$W_{I_{(t-N)}} = p(\theta|I_{(t-N)}) = \mathcal{N}(\mu_{t-N}, \Sigma_{t-N}) \quad (7)$$

The above two distributions can each be computed according to equation 3 where r, and F are substituted with the rewards and the item features of the recommendations in the corresponding window.

The distance between W_{I_t} and $W_{I_{(t-N)}}$ is used as a measure of dissimilarity between these two windows. Various measures such as KL-divergence can be used to compute the distance between the two windows. In our experiments, we used *Mahalanobis distance* which is computed as follows:

$$\Sigma = \frac{\Sigma_t + \Sigma_{t-N}}{2} \quad (8)$$

$$distance = (\mu_t - \mu_{t-N})^T \Sigma^{-1} (\mu_t - \mu_{t-N})$$

If the user feedback behavior is significantly different between the two windows, then the models for W_{I_t} and $W_{I_{(t-N)}}$ would be different. Therefore, the change detection module in our system is based on the heuristic that sudden changes in the users' feedback behavior would result in sudden changes of the distance measure computed in 8. Based on this heuristic, we compute the distance measure at each step t, and exploit a change point analysis approach to detect sudden changes in a user's feedback behavior.

In our system, we utilize the technique described in [16] for change point analysis. This procedure involves iteratively applying a combination of cumulative sum charts (CUSUM) and bootstrapping to detect the changes. For each of the detected changes, a confidence level is provided which represents the confidence of the predicted change point. Given a pre-specified threshold, only those change points that have confidence level above that threshold are accepted. Also, in our adaptation of this approach, we have incorporated a *look ahead (LA)* parameter to improve the accuracy. Our change detection module accepts a change point only if there are at least LA observations after the predicted change point.

In our system, we assume a user's interactions after the change point reflect his or her most recent preferences and the interactions before the change point reflect the user's preferences in a different context. The strategy to aggregate the preferences from different contextual states depends on the specific application in which the recommender is being used. In some applications, a user's preferences in a different context may still contain some useful information about the *general* preferences of that user. In this case, a good strategy for the recommender would be to consider all the user's preferences while giving more weight to the feedbacks in the current context. In other applications, a user's preferences in different contexts are almost independent. For example, the user may follow different independent tasks at different contextual states. In this situation, a good strategy would be to provide recommendations just based on the preferences corresponding to the current context of the user, or in other words, discarding the interactions before the change of context. The recommender presented in this paper, discards users' interactions before the last detected change point and produces the recommendations only based on the interactions in the current context.

Depending on the type of utility (binary, rating), the change point detection may falsely detect change points at earlier interactions where it starts to learn the users' preferences. To avoid this situation, we have incorporated another parameter in our model to avoid splitting the user profile if there are fewer interactions in the user's profile than the pre-specified *splitting threshold*.

In our future work, we plan to investigate adapting a bandit strategy that considers all of user's preferences. This would include, defining a utility function that while giving more weight to the preferences in the current context, also considers the interactions before the change point.

5. EVALUATIONS

Evaluation of contextual recommenders can be very challenging if contextual factors are unobservable. In this case, as explained in section 2, context cannot be represented as a set of known attributes. Instead, change of context is inferred from the change in users' activities and interactions with the system. There are various datasets based on representational view to context. However, there are very few publicly available datasets that explicitly contain users' change of preferences that could be used as the ground truth for evaluation if contextual factors are not known. In our evaluations, we designed two different experiments for simulating users' change of behaviors. In our first experiment, we used Yahoo user-artist ratings dataset where ratings are in the range of 0 to 100. To simulate sudden changes in a user's rating behavior, we built test hybrid profiles by merging two random user profiles. Treating the two selected user profiles as the rating behavior of a single hybrid user in two different contextual states, we evaluated our method in correctly predicting the change of context and producing context-aware recommendations.

In the second experiment, we used a Web navigational dataset containing users' browsing sessions. Assuming that a user's change of context is more likely to happen between sessions rather than within sessions, we evaluated our algorithm ability in correct detection of the transition between two sessions. We then compared our recommender with various non-contextual baselines.

5.1 Experiment I

The dataset used in our first experiment was the Yahoo! Music ratings of musical artists version 1.0. This dataset represents a snapshot of the Yahoo! Music community's preferences for various musical artists. It contains over ten million ratings of musical artists given by Yahoo! Music users over the course of a one month period sometime prior to March 2004. It contains ratings in the range of 0 to 100 given by 1948882 users to 98213 artists.

This experiment evaluated our algorithm in adapting to changes in users' rating behaviors that could be a result of contextual changes. For example, members of a household may use a single account to rate movies on a movie streaming application. Each member can rate movies differently and as a result, the *household hybrid* user may show significant differences in rating behavior. The purpose of this experiment was two-fold: first, we evaluated the accu-

Table 1: Method parameters in experiment I

Method	Parameters
Bandit recommender	Number of features = 5, $\mu_\theta = 0$, $\sigma_\theta = 0.5$, $\sigma_r = 0.5$
Contextual Recommender (proposed method)	Number of features = 5, $\mu_\theta = 0$, $\sigma_\theta = 0.5$, $\sigma_r = 0.5$, Change detection window size = 5, Change detection confidence = 0.95, Chage detection look ahead = 3, splitting threshold = 10
User-based kNN	Number of neighbors = 10

racy of our change detection approach in real-time detection of changes. Secondly, we examined the performance of our recommender compared with non-contextual methods. The evaluation was performed in the framework of an interactive recommender that interacts with the user in a number of consecutive rounds. In each round, the recommender presents a recommendation and observes the user's feedback (utility) for the recommended item. The recommenders were evaluated based on the average utility gained over the session of interaction.

We followed a five-fold cross validation approach for evaluation. In each round, one of the folds was used for testing, one was used for tuning the parameters and the remaining folds were used for training the model. To simulate the changes in a user's rating behavior, we randomly selected two users u_1 and u_2 from our test data and combined them to create a hybrid user u_h. We treated each of these two users as the rating behavior of a single hybrid user in two different contextual states.

To simulate an interactive recommendation process, at each step of interaction, the recommender used the ratings in the hybrid test user profile to recommend a new item that has not been presented to the user before. We randomly chose an item rating from the u_1 profile and used it as the initial profile of the hybrid user. For the first $T = 30$ rounds of interaction, the observed utility for each recommended item x, was assumed to be the rating assigned to x by the user u_1. We assumed that the contextual change occurs after T steps of interactions. Therefore, for the next $T = 30$ steps, the same process was repeated with the difference that the observed utility for a recommended item was set to the rating for that item in the profile of u_2. For this evaluation, we filtered the test users so that each user had at least T ratings in the user's profile.

If the user has not rated a recommended item, one option would to ignore the recommendation and discard it from the evaluation results. However, this would create a bias in the experiment. If the user rates none of the recommendations, then the recommender should be negatively scored (in comparison to a recommender that recommends items that at least are rated by the user). Setting the utility to zero for the non-rated items would also be too strict, as the user may not necessarily dislike the items that she hasn't rated yet. Another possibility would be to set the utility to average rating of all the users for that item. This also creates a bias as item popularities are not uniform. For example, if an algorithm suggests very rare items with high average ratings, it will gain a high utility score in the evaluation. A similar bias occurs if the average rating of the user is used as the utility for non-rated items. In our evaluation, if a recommended item had not been rated in the user's profile, the observed utility was set to the average rating of all items in the dataset.

The set of parameters used in this experiment for each of the algorithms are specified in table 1.

User-based kNN was used as one of our baselines for this experiment. Cosine similarity metric was used to compute the similarity of users. In each round of interaction, the kNN algorithm computed the item scores and recommended a new item that has the highest score.

The *optimal recommender* was chosen as another baseline to compare our method with an ideal algorithm. This recommender has full knowledge of users' preferences and the contextual changes. For the first $T = 30$ iterations, it followed a greedy strategy and recommended new items that had the highest rating in the u_1 profile. After change of context at $T = 30$, in each round it recommended a new item that had the highest rating in the u_2 profile. The standard bandit algorithm with Thompson sampling heuristic (as describe in section 4.2) was used as another baseline in our experiment and was trained with the parameters specified in table 1. In the remaining of the paper, we refer to this baseline as the standard bandit recommender.

Table 2 illustrates an example interaction session generated based on the standard bandit recommender for this experiment.

The recommendations that are not rated in the user's profile are marked as '-'. For each recommended artist, a few of most similar artists that have previously been presented to the user are also shown. The artist similarities are determined based on the Last.FM data. For any given artist, the Last.FM API can be used to find the most similar artists as well as the degree of similarity between the artists. In table 2, the similarity levels are abbreviated as M(medium), H(high), VH(very high), and SH(super high). As can be

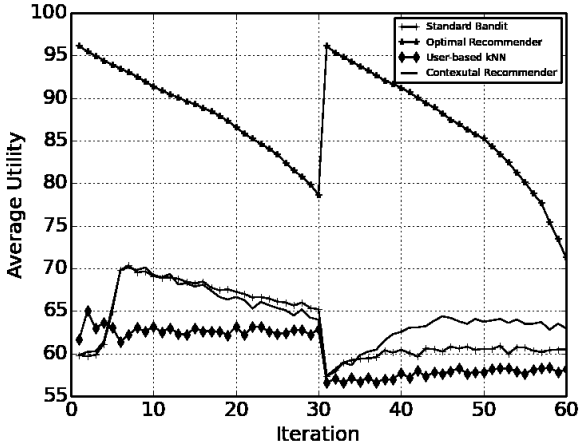

Figure 1: Experiment I-Average utility at each iteration

seen, most of the first few (5) recommendations are not rated by u_1. After iteration 6, the recommender starts generating better recommendations as most of them have high ratings in the u_1 profile. After the simulated change of context occurs at iteration 31, the user (u_2) has not rated most of the

Table 2: An example interaction session generated using the standard bandit recommender

Step	Artist Name	Utility	Similar artists	Step	Artist Name	Utility	Similar artists
1	Chingy	-		31	Foo Fighters	-	Nirvana(SH), Red Hot Chili Peppers(SH)
2	Disturbed	90		32	Hoobastank	-	Trapt(SH), 3 Doors Down(SH)
3	Led Zeppelin	-		33	System Of A Down	-	Korn(H), Disturbed(H)
4	Faith Hill	-		34	No Doubt	70	
5	Eminem	-		35	U2	-	Red Hot Chili Peppers(L)
6	Linkin Park	100	Disturbed(M), Eminem(M)	36	Alien Ant Farm	-	Incubus(SH),Puddle Of Mudd(SH)
7	Staind	100	Disturbed(H)	37	Creed	-	Staind(VH), 3 Doors Down(H)
8	Good Charlotte	-		38	Smile Empty Soul	-	Trapt(SH), Staind(VH)
9	Metallica	90	Led Zeppelin(H), Disturbed(H)	39	Slipknot	-	Korn(VH), System of a Down(VH)
10	Green Day	90	Good Charlotte(M), Linkin Park(L)	40	Michelle Branch	-	-
11	Red Hot Chili Peppers	90	Metallica(M)	41	The Ataris	-	Jimmy Eat World(VH)
12	The Offspring	-	Metallica(VH), Green Day(VH)	42	The White Stripes	-	Nirvana(M)
13	Korn	-	Disturbed(H), Metallica(H)	43	Avril Lavigne	-	
14	Evanescence	100	Disturbed(H), Linkin Park(H)	44	Dashboard Confessional	-	Jimmy Eat World(SH)
15	Blink 182	90	Good Charlotte(H), Green Day(H)	45	Matchbox Twenty	-	-
16	Limp Bizkit	50	Korn(SH), Disturbed(H)	46	New Found Glory	-	blink-182(SH)
17	Nickelback	100	Linkin Park(VH), Staind(VH)	47	Chevelle	-	Staind(VH), Trapt(H)
18	Trapt	90	Staind(VH), Disturbed(M)	48	Brand New	-	Jimmy Eat World(H)
19	Simple Plan	-	Green Day(VH), Blink 182(H)	49	Nine Inch Nails	-	-
20	Papa Roach	90	Nickelback(VH), Linkin Park(VH)	50	Switchfoot	-	
21	Nirvana	90	Metallica(H)	51	Goo Goo Dolls	-	3 Doors Down(H), Trapt(H)
22	Godsmack	100	Disturbed(SH), Staind(H)	52	Rob Zombie	-	Korn(M)
23	P.O.D.	100	Papa Roach(VH), Staind(VH)	53	Audioslave	-	Foo Fighters(SH), Incubus(VH)
24	All-American Rejects	90	Good Charlotte(SH), Blink 182(VH)	54	Sugarcult	-	Good Charlotte(VH)
25	Puddle Of Mudd	100	Staind(SH), Trapt(VH)	55	Deftones	-	Korn(VH), Limp Bizkit(H)
26	Sum 41	-	Blink 182(SH), Green Day(SH)	56	The Used	-	Blink-182(H), Papa Roach(H)
27	3 Doors Down	100	Nickelback(SH), Trapt(SH)	57	Lifehouse	-	3 Doors Down(SH)
28	Incubus	100	Staind(H)	58	Static-X	-	Korn(VH), Disturbed(VH)
29	A.F.I.	90	Good Charlotte(L)	59	Fountains Of Wayne	-	
30	Jimmy Eat World	90	Blink 182(VH), A.F.I(H)	60	Taking Back Sunday	-	Jimmy Eat World(VH)

recommendations, showing the dramatic degradation of the recommender performance. Note that many of the recommendations are artists similar to those rated highly by u_1. For example, Foo fighters recommended at iteration 31, is highly similar to Nirvana recommended at iteration 21 or Hoobastank, recommended at iteration 32, has super high similarity to 3 Doors Down, which has received utility of 100 at iteration 27. This example shows that a traditional bandit approach cannot adapt to the changes in the users' preferences.

Figure 1 presents the average utility at each iteration computed over the set of test users. The optimal recommender achieves the highest utility of 96 at the first step and its utility decreases as more iterations pass. This can be explained by the fact that in the first T iterations items are recommended in the decreasing order of their ratings in the u_1 profile, therefore the average utility gradually decreases. After iteration 30, the optimal baseline recommends items based on their ratings in the u_2 profile and so a jump in utility can be observed at iteration 31. Both the standard bandit recommender and the contextual recommender need 5-6 iterations to learn users' preferences and show a dramatic improvement in the utility. As indicated in table 1, the number of factors is set to 5 for these two methods. We repeat this experiment for different number of factors. The experiments revealed that by lowering the number of factors, it takes less iteration for the algorithm to reach its best performance. However, the highest achieved utility would be smaller as well.

Figure 2 illustrates the average distance computed as in equation 8 in each round of interaction. As can be seen in the figure, at iterations 33 to 35, distance has the maximum value. As the window size for change detection is set to 5, it takes at least 3 rounds until W_{I_t} mostly contains the interactions after the change of context and $W_{I_{t-N}}$ only has the interactions before the change of context and therefore, the distance metric become maximized.

Figure 3 presents the average number of change points detected at each iteration. As evident in the figure, the average frequency of change points reaches the maximum value of 0.5 at iteration 6 where $W_{I_{t-N}}$ has one interaction and W_{I_t} window contains 5 (window size) interactions. Setting

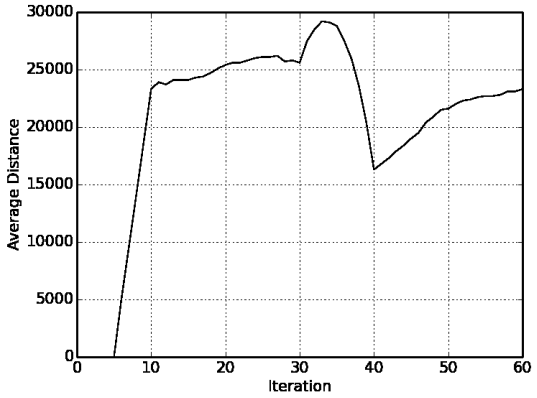

Figure 2: Expriment I-Average distance at each iteration

the splitting threshold to 10 ensures that the user profile is not split for the detected change points before round 10. Also, at iteration 38 there's another local maximum point where average frequency of change points reaches 0.4. This peak corresponds to the change of context in the hybrid user profile.

5.2 Experiment II

The data set used in our second experiment is the "CTI data", which is generated based on the server log data from the host Computer Science department. The site contains various online applications, including admissions application, online advising, online registration, and faculty-specific Intranet applications. The dataset contains 20950 sessions. We pre-processed the data to remove pages with frequency of one in the whole dataset and also the top 30 most frequent pages. After pre-processing the data, we identified 5319 users, and 2453 distinct pageviews. Following a 10-fold cross-validation approach, the data was randomly split into ten folds. In each run, one of the folds was used for testing, one was used for tuning the parameters and the remaining folds were used for training the model. For evaluation, we further filtered the test cases to include only those

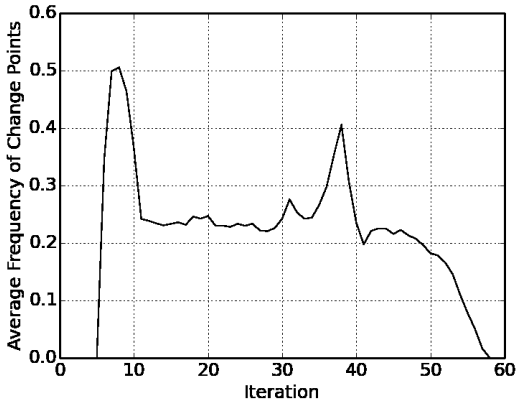

Figure 3: Experiment I-Average frequency of change points at each iteration

users that have at least two sessions in their profiles each containing at least L pageviews where L is the rounded up average session size and equals to seven in our dataset. For each test user, we randomly chose two sessions s_1 and s_2 (with the minimum length of L) and discarded the remaining sessions. We chose the first pageview in s_1 and used it as the initial profile (or entry page) of the user. To simulate an interactive recommendation process, for the first $T = 10$ iterations, the recommender used the current pageviews in the user's profile to recommend an item that she had not yet viewed. If s_1 contained the recommended page, then it was counted as a *hit*, and otherwise, it was counted as a *miss*. The recommended item and the corresponding binary feedback (miss or hit) were then added to the user's profile. After $T = 10$ iterations, we assumed the user starts a new session, s_2. For the new session, the same interactive recommendation process was repeated for another T iterations with the difference that if the recommended page was included in s_2, it was counted as a hit, and otherwise a miss. Note that the user profile in iterations between T and $2T$ contains the recommended items from both sessions s_1 and s_2.

The recommendation approaches were evaluated based on average Click Through Rate (CTR) obtained in each iteration across all the test users. CTR at iteration t equals to the total number of hits at step t divided by the total number of recommendations at that step.

We used *First Order Markov Model* as one of our baselines. This approach models each page as a state and the state transition probabilities are estimated from the navigational training data. It generates recommendations based on the state transition probability of the last page in the active user session.

The standard bandit recommender with Thompson sampling heuristic was used as another baseline in our approach. The set of parameters used for each of evaluated algorithms are specified in table 3. Figure 4 indicates the average CTR over the set of all the test users at different interaction steps for each of the recommendation algorithms. The CTR value for both the standard bandit and the contextual recommenders almost increases at each step for the first 7 iterations. The CTR reaches its maximum at the seventh iteration and has value of 0.31, and 0.32 for the contextual recommender and the standard bandit recommender respectively. Similar to experiment I, the number of iterations it

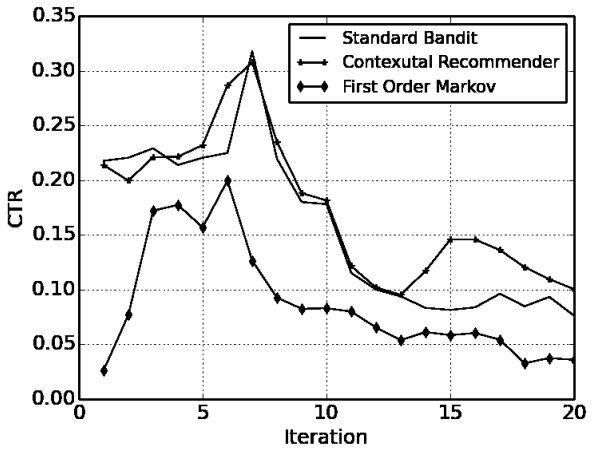

Figure 4: Experiment II-Click through rate at different iterations

takes for the algorithm to reach the maximum utility peak almost equals to the number of features. In our experiments, we tuned the parameters to maximize the average CTR.

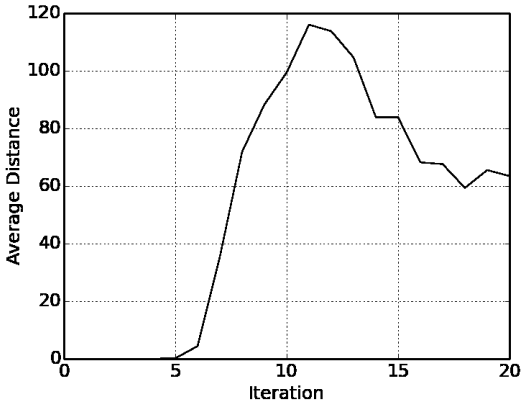

Figure 5: Experiment II-Average distance at each iteration

Note that the popularity of the pages is not uniform in our dataset. Some of the pages are more popular and have higher prior of being visited by the users. As the recommender recommends only new pages, the CTR gradually starts decreasing as the popular pages are recommended.

The contextual recommender and the standard bandit recommender both show very similar CTR values for the first 10 iterations with the contextual recommender having a slightly better performance after the third iteration. Also, after change of context at iteration 11, the CTR decreases with a higher rate (in comparison to the iteration 10). After change of context, the difference between the CTR achieved by these two algorithms starts increasing. At iteration 15, the CTR for contextual recommender is close to 0.15 while it is only 0.08 for the standard bandit recommender.

In comparison to both the bandit and contextual recommenders, Markov model has much smaller CTR at all the iterations. For Markov Model recommender, CTR mostly increases in the first 6 iterations and reaches to its maximum peak of 0.2 at the sixth iteration. The CTR then decreases till the last iteration.

Table 3: Method parameters in experiment II

Method	Parameters
Bandit recommender	Number of features = 6, $\mu_\theta = 0$, $\sigma_\theta = 1$, $\sigma_r = 0.001$
Contextual Recommender	Number of features = 6, $\mu_\theta = 0$, $\sigma_\theta = 1$, $\sigma_r = 0.001$, Change detection window size = 3, Change detection confidence = 0.99, Change detection look ahead = 0, splitting threshold = 3

Figure 5 indicates the average distance measure at each iteration, computed by the change detection module. Note that the average distance has its peak value right after change of context (at iteration 11). Figure 6 indicates the average frequency of change points at each iteration. For this dataset, choosing a high confidence threshold value helps to avoid false detection of change points at the earlier iterations. The average detected change points have its maximum at iteration 13 corresponding to the change of context.

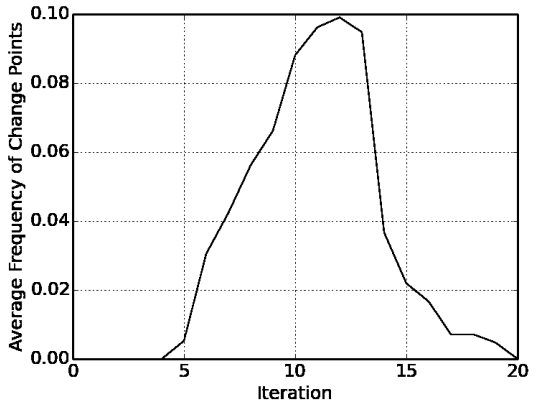

Figure 6: Experiment II-Average frequency of change points at each iteration

6. CONCLUSIONS

In this paper we have presented a novel approach for adaptation to contextual changes in interactive recommender systems. In an online recommendation scenario, at each step the system provides the user with a list of recommendations and receives feedback from the user on the recommended items. Various multi-armed bandit algorithms can be used as recommendation strategies to maximize the average utility over each user's session. Our approach extends the existing bandit algorithms by considering the sudden variations in the user's feedback behavior as a result of contextual changes. Our system contains a change detection component that determines any significant changes in the users' preferences. If a change is detected, the bandit algorithm based on Thompson sampling, prioritizes the observed feedback after the detected change point reflecting the preferences of the user in the current context. The proposed recommender, discards users' interactions before the last detected change point and produces the recommendations just based on the interactions in the current context. In our future work, we plan to investigate various approaches to take into account the impact of users' interactions in prior contexts on the current user behavior. One approach might involve defining a utility function that while giving more priority to the interactions in the current context, would also take into account user's feedback before the change point.

7. REFERENCES

[1] G. Adomavicius, B. Mobasher, F. Ricci, and A. Tuzhilin. Context-aware recommender systems. *AI Magazine*, 32(3):67–80, 2011.

[2] S. Agrawal and N. Goyal. Thompson sampling for contextual bandits with linear payoffs. volume 28 of *JMLR Proceedings*, pages 127–135, 2013.

[3] C. Baccigalupo. *Poolcasting: an intelligent technique to customise music programmes for their audience*. PhD thesis, Universitat Autonoma de Barcelona, 2009.

[4] D. Bouneffouf, A. Bouzeghoub, and A. L. GanÃğarski. A contextual-bandit algorithm for mobile context-aware recommender system. In *ICONIP (3)*, volume 7665 of *Lecture Notes in Computer Science*, pages 324–331. Springer, 2012.

[5] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. In *NIPS*, pages 2249–2257, 2011.

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

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

[8] N. Hariri, B. Mobasher, and R. D. Burke. Query-driven context aware recommendation. In *RecSys*, pages 9–16. ACM, 2013.

[9] M. Kaminskas, F. Ricci, and M. Schedl. Location-aware music recommendation using auto-tagging and hybrid matching. In *RecSys*, pages 17–24. ACM, 2013.

[10] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW*, pages 661–670. ACM, 2010.

[11] O. Meyers. A mood-based music classification and exploration system. Master's thesis, Massachusetts Institute of Technology, 2007.

[12] K. P. Murphy. *Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series)*. The MIT Press, 2012.

[13] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

[14] P. Rusmevichientong and J. N. Tsitsiklis. Linearly parameterized bandits. *Math. Oper. Res.*, 35(2):395–411, 2010.

[15] S. L. Scott. A modern bayesian look at the multi-armed bandit. *Appl. Stochastic Models Bus. Ind.*, 26(6):639–658, Nov. 2010.

[16] T. Wayne. Change-point analysis: a powerful new tool for detecting changes. 2000.