Clustering Based Online Learning in Recommender Systems: A Bandit Approach

Linqi Song, Cem Tekin, Mihaela van der Schaar

Electrical Engineering Department, UCLA
Email: songlinqi@ucla.edu, cmtkn@ucla.edu, mihaela@ee.ucla.edu

ABSTRACT

A big challenge for the design and implementation of large-scale online services is determining what items to recommend to their users. For instance, Netflix makes movie recommendations; Amazon makes product recommendations; and Yahoo! makes webpage recommendations. In these systems, items are recommended based on the characteristics and circumstances of the users, which are provided to the recommender as contexts (e.g., search history, time, and location). The task of building an efficient recommender system is challenging due to the fact that both the item space and the context space are very large. Existing works either focus on a large item space without contexts, large context space with small number of items, or they jointly consider the space of items and contexts together to solve the online recommendation problem. In contrast, we develop an algorithm that does exploration and exploitation in the context space and the item space separately, and develop an algorithm that combines clustering of the items with information aggregation in the context space. Basically, given a user's context, our algorithm aggregates its past history over a ball centered on the user's context, whose radius decreases at a rate that allows sufficiently accurate estimates of the payoffs such that the recommended payoffs converge to the true (unknown) payoffs. Theoretical results show that our algorithm can achieve a sublinear learning regret in time, namely the payoff difference of the oracle optimal benchmark, where the preferences of users on certain items in certain context are known, and our algorithm, where the information is incomplete. Numerical results show that our algorithm significantly outperforms (over 48%) the existing algorithms in terms of regret.

Index terms-- Recommender systems; online learning; clustering algorithms; multi-armed bandit.

1. INTRODUCTION

With the rapid growth of online web services, a huge number of items become available to users [1], such as movies at Netflix, products at Amazon, webpages at Yahoo!, and advertisements at Google. Most widely used recommender systems, such as video and audio recommender systems [5][8], have very large item sets. The goal of such recommender systems is to assist its users in finding their preferred items from the large set of items [1][2].

The preference of a user on a particular item is learned through a random payoff, which is received by the recommender system based on the response of the user to the recommendation. For example, in the movie recommendations, the payoffs are the rating scores (e.g., 1 to 5) on movies rated by the users; in the webpage recommendations, the payoffs are measured by the users' click behaviors (i.e., 1 for a click and 0 for no clicks).

Two popular recommendation approaches are filtering-based and machine learning-based techniques [3]. Filtering-based approaches, such as collaborative filtering [4][5], content-based filtering [2][6] and hybrid approaches [7][8], employ the historical data of users' feedback to calculate the future payoffs of users based on some prediction functions.

Machine learning-based methods, such as *Multi-Armed Bandit (MAB)* algorithms [20]-[23] and *Markov Decision Processes (MDPs)* algorithms [24], use machine learning techniques to solve the recommendation problem. MDP-based learning approaches model a fraction of the historical data of a user as the state and the possible items as the action set, and try to maximize the long-term total payoff [24]. However, a key disadvantage of such MDP approaches is that the state set will grow fast as the number of items increases, thereby resulting in very slow convergence rates. MAB-based approaches [9]-[14], such as ε - greedy [14] and UCB1 [12], provide not only asymptotic convergence to the optimum, but also a bound on the rate of convergence for any time step. They do this by balancing exploration and exploitation, where exploration means recommending different sets of items to learn about their expected payoffs, and exploitation means recommending the best set of items based on the observations made so far.

However, in many applications, such as movie recommendations, it may not be sufficient to only consider the user-item space. It is also important to incorporate *context* into the process in order to improve the quality of the recommendations to users based on specific circumstances [15]-[17]. Such context-aware recommender systems have been recently studied [15]-[23]. For example, in movie recommendations [17], it is also important to consider the time when a movie should be seen (e.g., weekdays, weekends or special days such as Valentine's Day), the location in which the movie should be seen, (e.g., home or movie theater), the companion with which the movie is seen (friends, family, alone, co-workers, etc.). Thus, incorporating contexts extends the user-item space used traditionally for recommender systems to the user-item-context space, in order to evaluate the payoffs of recommendations. Moreover, if users have privacy concerns, the recommender system cannot recognize a particular user when the user arrives. In this case, the users' features (cookies) can be considered as contexts. In [20][21], the payoff functions are assumed to be linear in contexts, and an algorithm named LinUCB is proposed to solve the news article recommendation problem with contexts. However, the linear payoff assumption is not always true in practice. In a more general setting, it is assumed that the context space is a bounded metric space with a Lipschitz condition, and recommendation algorithms are proposed based on the context space partition. These works [15]-[23], however, do not take into account the large item space, which is a key challenge in practice.

Another strand of related works studies recommender systems in which the item space is large [3][4]. MAB-based learning algorithms for large item space have been considered in [25][26]. In [26], the item space has been partitioned into finite number of subspaces, and the learning is performed on the subspace level

The material is based upon work funded by the US Air Force Research Laboratory (AFRL). Any opinions, findings, and conclusions or recommendations expressed in this article are those of the authors and do not reflect the views of AFRL.

instead of the item level. Alternatively, in [25], an item cluster tree is considered, and tree search methods are performed to find the optimal item. In contrast, we consider a finite but large item space, based on which the learning is performed on the cluster level instead of item level. Moreover, contexts are not considered in these works [3][4][25][26].

Building contextual recommender systems with a large item space has become challenging. In [27], the recommendation is performed by merging the context space $\mathcal{X}$ and item space $\mathcal{I}$ into a joint space $\mathcal{X} \times \mathcal{I}$, and by partitioning of the new space. However, this work needs to know the Lipschitz constant in the algorithm, which is difficult to implement in practice. Furthermore, the merging of spaces greatly increases the space dimension (i.e., the dimension of $\mathcal{X}$ plus the dimension of $\mathcal{I}$), resulting in a slow convergence rate.

In this paper, we consider the design and implementation of recommendation systems which consider the users' contextual information and have a large number of items. For this, we propose a contextual bandit approach based on the item cluster tree. To the best of our knowledge, our work is the first to solve the recommendation problem using an item cluster tree based contextual bandit approach. Firstly, different from [27], we separately consider the context and item spaces. We model the context space as a general continuous and bounded space, although our results still hold for a finite context space. Since items can always be categorized as a cluster tree for different online services, we can use a general cluster tree structure to model the item space, and form a set of clusters based on this structure. Secondly, our algorithm selects item clusters based on past observations in a dynamic subspace in the context space each time, in contrast to existing works [22][23][25]-[27], where only the past observations in static or semi-dynamic subspaces are used. Thirdly, our algorithm does not need to know the Lipschitz constant, which is only used to evaluate the performance of the algorithm. A detailed comparison with the existing works is presented in Table I.

In our model, the leaves of the tree represent the items, and the nodes of the tree represent the clusters (a leaf can also be seen as a cluster with only one item). The metric (distance) to evaluate the *similarity* between items is defined based on this cluster tree. In the proposed algorithm, we model the possible selections (referred to as the arms) of the MAB model as a layer of nodes (clusters) at depth-*d* of the tree, instead of each single item, which reduces the number of possible selections and hence, increases the learning speed.

The proposed algorithm works in discrete time periods and at each time period it alternates between an *exploration phase* and an *exploitation phase* depending on the past selections. Each time, a user with a specific context arrives, and the recommender system aggregates information over a ball (referred to as the *active ball*) centered on the current context that contains a sublinear number of past arrivals. Then, if the times that a cluster is selected in the active ball up to now is below a certain threshold, the current period will be an exploration phase and that cluster will be selected; otherwise, the period will be an exploitation phase and the current "best" cluster (in terms of sum average payoffs) will be selected. Then, a randomly selected item in the selected cluster is recommended to the user. When the selection is made, a random payoff is observed by the recommender system. The goal of learning is to minimize the difference (referred to as the *regret*) between the optimal expected total payoffs that can be achieved if all expected payoffs are known and the expected total payoffs gained through our learning algorithm, in which the information is incomplete.

Table I Comparison between the Proposed and Existing Solutions

	Context	**Dependent items**	**Item cluster tree**	**Context subspace**	**Linear payoff**
[13]	No	No	No	No	Yes
[20][21]	Yes	No	No	No	Yes
[22][23]	Yes	No	No	Partition	No
[25]	No	Yes	Yes	No	No
[26]	No	Yes	Yes	No	No
[27]	Yes	Yes	No	Partition	No
Our work	Yes	Yes	Yes	Dynamic subspace	No

2. SYSTEM MODEL

2.1. Recommender System

A recommender system consists of a set of items, denoted by $\mathcal{I} = \{1,2,\cdots,I\}$. In the item space, the similarity distance is defined as a metric $s_I : \mathcal{I} \times \mathcal{I} \to \mathbb{R}$, which is based on the features of the items and known to the recommender system. A smaller $s_I(i,i')$ implies that two items i and i' are more similar. We denote the context set by $\mathcal{X} = [0,1]^{d_C}$. Each context $\boldsymbol{x} \in \mathcal{X}$ is a d_C dimensional vector, i.e., $\boldsymbol{x} = (x_1, x_2, \cdots, x_{d_C})$ and each component of $\boldsymbol{x}$ is a real number in $[0,1]$. We denote the metric in context space by $s_C : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. A smaller $s_C(\boldsymbol{x},\boldsymbol{x}')$ implies that the two contexts $\boldsymbol{x}$ and $\boldsymbol{x}'$ are more similar.

The recommender system operates in discrete time slots $t = 1,2,3\cdots$. The context arrival is independent identically distributed (i.i.d.), and each time it is sampled from a fixed but unknown distribution. We assume that the probability density $f(\boldsymbol{x})$ of $\boldsymbol{x}$ satisfies:

$$f_{\min} \le f(\boldsymbol{x}) \le f_{\max}, \forall \boldsymbol{x} \in \mathcal{X} \ . \quad (1)$$

This bounded probability density indicates that there will be a positive arrival probability for a user with any context.

For a user with context $\boldsymbol{x} \in \mathcal{X}$, the payoff of choosing item i is denoted by $r_{i,\boldsymbol{x}} \in [0,1]$, which is a random variable drawn from a fixed but unknown distribution and its average payoff is denoted by $\mu_{i,\boldsymbol{x}}$. Only the payoffs of the recommended items can be observed by the recommender system and can be used for further recommendations.

Note that we will add the subscript t to context $\boldsymbol{x}$ when referring to the learning process over each period. The sampled $\boldsymbol{x}_t$ can be observed by the recommender system when the user arrives. The components of context and payoffs in [0, 1], are just for notational simplicity and in general they can be in any bounded interval. In the Euclidean space $\mathcal{X}$, the metric $s_C(\boldsymbol{x},\boldsymbol{x}')$ can be any Euclidean norm.

In the context space, an item has similar payoffs when similar contexts arrive; we formalize this in terms of a Lipschitz condition as follows.

Assumption 1 (Lipschitz condition for contexts): Given two contexts $\boldsymbol{x},\boldsymbol{x}' \in \mathcal{X}$, we have the following assumption: $| \mu_{i,\boldsymbol{x}} - \mu_{i,\boldsymbol{x}'} | \le L_C s_C(\boldsymbol{x},\boldsymbol{x}')$, for any item i .

In the item space, the expected payoffs of similar items are similar, given the same context. We formalize this as follows.

Assumption 2 (Lipschitz condition for items): Given two items $i, i' \in \mathcal{I}$, we have the following assumption: $|\mu_{i,\boldsymbol{x}} - \mu_{i',\boldsymbol{x}}| \le L_I s(i,i')$, for any context $\boldsymbol{x}$.

2.2. Item cluster tree

Items in the system are often categorized as a cluster tree [27]. Recall that in a cluster tree, each leaf represents an item and each node represents a cluster.

We define all the nodes at depth $d \ge 0$ as *layer d* and the node/cluster in layer *d* is denoted by $C_{d,l}$, where $l \in \{1,2,\cdots L_d\}$ and L_d is the number of nodes at depth *d*. The diameter of a cluster $D_{C_{d,l}}$ is defined as the maximal distance of the items in that cluster, namely, $D_{C_{d,l}} = \max_{i,i' \in C_{d,l}}\{s(i,i')\}$. In a cluster tree structure, a general tree metric can be defined as the mapping from depth of the node in the tree to the diameter bound of the cluster, namely, $s_T : \mathbb{N} \to \mathbb{R}$. Thus, $s_T(d)$ denotes the diameter bound of the clusters at depth *d*, namely, $D_{C_{d,l}} \le s_T(d)$ for any cluster l at depth *d*. For a tree metric, the diameter bound of cluster at depth *d* is not smaller than that at depth *d*+1, i.e., $s_T(d) \ge s_T(d+1)$.

There are several examples of the tree metric. Similar to [27], we use the *exponential tree metric*: $s_T(d) = \varepsilon^d$, where $\varepsilon \in (0,1)$ is the constant base for depth *d*.

Note that the advantage of applying this item cluster tree is to categorize items based on their similar features. For example, in some applications, the similarity between two items is hard to define, but the categories which the items belong to are easy to recognize. In this case, the tree metric is the appropriate evaluation of similarity in item space. Moreover, given the tree metric, an arbitrary metric can be formed into a cluster tree, which fulfills the tree metric [28]. Thus, for an item cluster tree, the Lipschitz condition in item space of Assumption 2 can be restated as Assumption 3.

Assumption 3 (Lipschitz condition for item cluster tree): For two items in the same cluster at depth *d*, namely, $i, i' \in C_{d,l}$, we have the following assumption: $|\mu_{i,\boldsymbol{x}} - \mu_{i',\boldsymbol{x}}| \le L_I s_T(d)$, for any context $\boldsymbol{x}$.

Note that the cluster tree can be partitioned by several nodes, which contain all the items and no two nodes contain the same item. Thus any item is included in one of the clusters. Moreover, there is a tradeoff between the number of clusters we need to explore and the cluster depth. Choosing clusters with larger depth will make more specific recommendations to the users, but it will also require learning more about specific characteristics of items and users.

2.3. Problem Formulation

In the contextual MAB based recommendation problem, the cluster-based partition $\mathcal{K}$ of the item space (i.e., a disjoint set of $K = |\mathcal{K}|$ clusters that cover the whole item space) is given to the recommender system. Each time, a user with context $\boldsymbol{x}$ arrives, a cluster $k \in \mathcal{K}$ is selected; an item i in that cluster is randomly recommended; and the payoff $r_t = r_{i,\boldsymbol{x}}$ is observed. Given the number of items in cluster k, M_k, the average payoff of cluster k equals $\mu_{k,\boldsymbol{x}} = \sum_{i \in k} \mu_{i,x} / M_k$. In order to make a recommendation, the algorithm should choose a cluster and an item from the chosen cluster. Therefore the algorithm needs to keep track of estimated performance of each cluster.

The recommender system selects the cluster at time *t* based on the current context and the history, which is a collection of past contexts, cluster selections, and payoff observations. The history is written as $h_t = \{(\boldsymbol{x}_1, k_1, r_1), (\boldsymbol{x}_2, k_2, r_2), \cdots, (\boldsymbol{x}_{t-1}, k_{t-1}, r_{t-1})\}$ for $t > 1$ and $h_1 = \varnothing$ for $t = 1$. We denote the history set of each period by $\mathcal{H}$, then the algorithm π is defined as a mapping from the current context and history to the recommendation action in time period *t*, namely $\pi : \mathcal{X} \times \bigcup_{t=1}^{\infty} \mathcal{H}^t \to \mathcal{K}$. We denote the set of all history-based algorithms by Π.

Thus, given the cluster set, the learning goal is to find an algorithm that maximizes the total average payoff, denoted by $U_\pi(T)$ (i.e., $U_\pi(T) = \sum_{t=1}^{T}[\mu_{\pi(\boldsymbol{x}_t,h_t),\boldsymbol{x}_t}]$), for any *T*:

$$\max_{\pi \in \Pi} U_\pi(T) = \max_{\pi \in \Pi} \sum_{t=1}^{T} [\mu_{\pi(\boldsymbol{x}_t,h_t),\boldsymbol{x}_t}] \qquad (2)$$

If all the information is known, the best choice is to choose $\pi^*(\boldsymbol{x}_t, h_t) = \arg\max_{k \in \mathcal{K}}[\mu_{k,\boldsymbol{x}_t}]$. However, in practice, the average payoffs $\mu_{k,\boldsymbol{x}_t}$ and the distribution of context $\boldsymbol{x}_t$ are not known. To measure the difference between $U_{\pi^*}(T)$ and $U_\pi(T)$, we alternatively define the regret of learning algorithm π as

$$R_\pi(T) = U_{\pi^*}(T) - U_\pi(T). \qquad (3)$$

Therefore, the design goal of the learning algorithm is to minimize the regret $R_\pi(T)$.

3. PROPOSED RECOMMENDATION ALGORITHM

In this section, we propose the *One-Layer Clustering Recommendation (OLCR)* algorithm, and prove that the regret of this algorithm is sublinear in *T*.

We first present an intuitive illustration of the algorithm in Fig. 1 and 2. Due to the huge number of items in the system, it is inefficient to compare against all the items in the system to select an appropriate one, since the learning speed will be very slow. One way to select the cluster set $\mathcal{K}$ is to choose clusters in the same layer, namely nodes with the same depth in the cluster tree. We formalize this selection in our OLCR algorithm, shown in Table II. Each time, the algorithm alternates between two phases: the *exploration phase* and the *exploitation phase,* depending on the history and current context. We denote by $B(x,\rho)$ the ball in context space with center x and radius ρ. We consider the $\lfloor t^\alpha \rfloor$ [1] ($0 \le \alpha \le 1$) closest past arrivals in context space and use a ball $B(x_t, \rho_t)$ to denote this subspace in the context space, as shown in Fig. 1. Let $N_{k,B(x_t,\rho_t)}$ denote the number of past selections of cluster k within the ball and $\overline{r}_{k,B(x_t,\rho_t)}$ denote the sum average payoffs of cluster *k* within the ball. The algorithm checks if there is any cluster whose past number of selections within the ball does

[1] $\lfloor y \rfloor$ denotes the maximal positive integer number that is smaller than or equal to y.

not exceed the threshold $At^{\beta}\ln t^{\alpha}$ ($0<\beta<\alpha$, $A>0$), and if so it explores that cluster by selecting it. If all the clusters are explored enough, i.e., past number of selections within the ball exceeds the threshold, it selects (exploits) the current best cluster (by comparing the sum-average of past payoffs of each cluster within the ball. When the selection is made, a random payoff r_{k,x_t} is observed by the recommender system.

The performance of the OLCR algorithm, in terms of regret up to time *T,* is given in Theorem 1.

Theorem 1: The regret $R_{OLCR}(T)$ up to time T of the OLCR algorithm is bounded by

$$R_{OLCR}(T) \le CT^{1-(\alpha-\beta)}\ln T + \frac{\pi^2}{3}(K-1) + \frac{2T^{1-\beta/2}}{\sqrt{A\alpha}(1-\beta/2)} + \frac{2L_1C_0d_cT^{(d_c-1+\alpha)/d_c}}{d_c-1+\alpha}, \qquad (4)$$

where C_0 is a constant, such that $E[\rho_t] \le C_0 t^{-(1-\alpha)/d_C}$, and $C = 1+2KA\alpha f_{\max} c_{d_C} C_{d_C}$ is a constant, such that c_{d_C} is the covering constant[2], and $C_{d_C} = \pi^{d_C/2}/\Gamma(d_C/2+1)$, $\Gamma(\cdot)$ is the Gamma function.

Proof: The proof of Theorem 1 is given in [29].

Note that the regret is sublinear in *T*, resulting in a sublinearly vanishing time average performance loss, which is a stronger result than the asymptotic convergence.

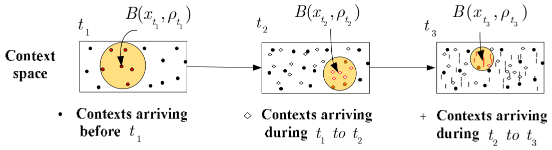

t_1
$B(x_{t_1},\rho_{t_1})$
t_2
$B(x_{t_2},\rho_{t_2})$
t_3
$B(x_{t_3},\rho_{t_3})$
Context space
• Contexts arriving before t_1
◇ Contexts arriving during t_1 to t_2
+ Contexts arriving during t_2 to t_3

Fig. 1. Dynamic subspace selection in the context space

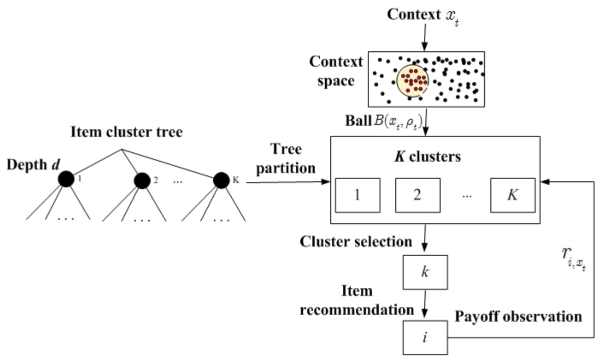


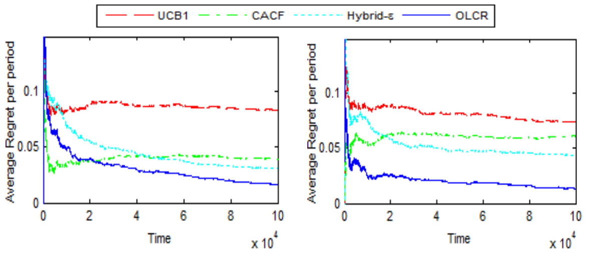
Fig. 2. Recommender system based on OLCR algorithm

4. NUMERICAL RESULTS

In this section, we compare the proposed OLCR algorithm with the traditional context-free UCB1 algorithm [12], the context-aware collaborative filtering (CACF) algorithm [18], and the hybrid-ϵ-greedy algorithm [19]. In the simulation, we consider a binary item cluster tree, whose metric fulfills the *exponential tree metric.*

TABLE II. One-Layer Clustering Recommendation Algorithm

```
1: Input: clusters $\mathcal{K}=\{1,2,\cdots K\}$, periods $T$.
2: for $t=1:T$ do
3: Observe the context $x_t$. Find a ball $B(x_t,\rho_t)$ with minimum radius $\rho_t$, which contains $\lfloor t^{\alpha}\rfloor$ of past arrived contexts.
4: if $\exists k,\ st\ N_{k,B(x_t,\rho_t)} \le At^{\beta}\ln t^{\alpha}$ then (Exploration Phase)
5:     select arm k, and randomly recommend an item in arm k.
6: else (Exploitation Phase)
7:     select $k=\arg\max_{k'\in\mathcal{K}}[\overline{r}_{k',B(x_t,\rho_t)}]$, and randomly recommend an item in cluster k.
8: end if
9: Receive the reward: $r_t$.
10: end for
```



(a) *K*=64 (b) *K*=16

Fig. 3. The regret performance comparison

Table III Comparison of Regrets up to *T*=100, 000

(K=64) / (K=16)	**UCB1**	**CACF**	**Hybrid-ϵ**	**OLCR**
Regret	8387 / 7422	3944 / 6134	3129 / 4385	1632 / 1362
Performance gain over others	81% / 82%	59% / 78%	48% / 69%	-

Simulation results are shown in Fig. 3 and Table III. We compare the average regret per period ($R_{\pi}(t)/t$ for algorithm π) of algorithms in Fig. 3, and show the comparison of regrets up to *T*=100, 000 in Table III. We can see that the OLCR algorithm significantly outperforms the context-free UCB1 algorithm, the CACF algorithm and the hybrid-ϵ-greedy algorithm, with 81%, 59% and 48% reduction of regret for *K*=64, and 82%, 78% and 69% reduction of regret for *K*=16, respectively. We can also see that the convergence rate of the OLCR algorithm increases when the number of clusters decreases from *K*=64 to *K*=16.

5. CONCLUSIONS

In this paper, we propose a contextual MAB based clustering algorithm to design and deploy recommender systems, in which both the contexts and the large item space are considered. To improve the learning speed, we consider partitioning the item cluster tree into a set of clusters. The algorithm alternates between the exploration and exploitation phases and aggregates the contextual information from a sublinear number of past arrived contexts. Theoretical results show that the algorithm can achieve a sublinear regret in time *T*, which is a stronger result than asymptotic convergence. Simulations show that our algorithm significantly outperforms the existing state-of-the-art algorithms by over 50%, in terms of regret.

[2] We use a set of balls with any radius ρ to cover a d_C dimensional space $\mathcal{X}$. If the distance between any two centers of the balls is greater than ρ, then the covering constant c_{d_C} is the constant, such that the maximum number of balls in the set is not greater than $c_{d_C}\rho^{-d_C}$.

6. REFERENCES

[1] P. Resnick and H. R. Varian, "Recommender systems," *ACM Comm.,* vol. 40, no. 3, pp. 56-58, 1997.

[2] M. Balabanovi and Y. Shoham, “Fab: content-based, collaborative recommendation,” *ACM Comm.*, vol. 40, pp. 66-72, 1997.

[3] G. Adomavicius and A. Tuzhilin, “Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions,” *IEEE Trans. Knowl. Data Eng.*, vol. 17, pp. 734-749, 2005.

[4] X. Su and T. M. Khoshgoftaar, “A survey of collaborative filtering techniques,” *Advances in AI,* vol 4, 2009.

[5] F. Sanchez, M. Alduan, F. Alvarez, J. Menendez, and O. Baez, “Recommender system for sport videos based on user audiovisual consumption,” *IEEE Trans. Multimedia*, vol. 14, no. 6, pp. 1546-1557, 2012.

[6] M. J. Pazzani and D. Billsus, “Content-based recommendation systems,” in *The Adaptive Web*, *LNCS*, vol 4321, pp. 325-341. Springer Berlin Heidelberg, 2007.

[7] R. Burke, “Hybrid recommender systems: Survey and experiments,” *User modeling and user-adapted interaction*, vol 12, no. 4, pp. 331-370, 2002.

[8] K. Yoshii, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno, “An efficient hybrid music recommender system using an incrementally trainable probabilistic generative model,” *IEEE Trans. Audio, Speech, Language Process.,* vol. 16, no. 2, pp. 435-447, 2008.

[9] H. Liu, K. Liu, and Q. Zhao, “Logarithmic weak regret of non-Bayesian restless multi-armed bandit,” in *IEEE ICASSP,* pp. 1968-1971, 2011.

[10] W. Dai, Y. Gai, B. Krishnamachari, and Q. Zhao, “The non-Bayesian restless multi-armed bandit: A case of near-logarithmic regret,” in *IEEE ICASSP,* pp. 2940-2943, 2011.

[11] K. Wang and L. Chen, “On optimality of myopic policy for restless multi-armed bandit problem: an axiomatic approach,” *IEEE Trans. Signal Process.*, vol. 60, no. 1 pp. 300-309, 2012.

[12] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time Analysis of the Multi-armed Bandit Problem," Machine Learning, vol. 47, pp. 235-256, 2002.

[13] Y. Deshpande and A. Montanari, “Linear bandits in high dimension and recommendation systems,” in *Proc. 50th Allerton Conference,* pp. 1750-1754, 2012.

[14] N. C.- Bianchi and G. Lugosi, *Prediction, learning, and games*. Cambridge Univ. Press, 2006.

[15] G. Adomavicius and A. Tuzhilin, “Context-aware recommender systems,” in *Recommender Systems Handbook*, Springer US, pp. 217-253, 2011.

[16] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin, “Incorporating contextual information in recommender systems using a multidimensional approach,” *ACM Trans. Inf. Syst.*, vol 23, no. 1, pp. 103-145, 2005.

[17] A. Said, S. Berkovsky, E. W. De Luca, and J. Hermanns, “Challenge on context-aware movie recommendation: CAMRa2011,” in *Proc. 5th ACM Conf. on Recommender Syst.*, pp. 385-386, 2011.

[18] A. Chen, “Context-aware collaborative filtering system: Predicting the user’s preference in the ubiquitous computing environment,” in *Location-and Context-Awareness*, Springer Berlin Heidelberg, pp. 244-253, 2005.

[19] D. Bouneffouf, A. Bouzeghoub, and A. L. Gançarski, “Hybrid-ε-greedy for mobile context-aware recommender system,” *Advances in Knowledge Discovery and Data Mining*, Springer Berlin Heidelberg, pp. 468-479, 2012.

[20] L. Li, W. Chu, J. Langford, and R. E. Schapire, “A contextual-bandit approach to personalized news article recommendation,” in *Proc. 19th WWW*, Raleigh, North Carolina, USA, 2010.

[21] W. Chu, L. Li, L. Reyzin, and R. E. Schapire, “Contextual bandits with linear payoff functions,” in *Int. Conf. AI Statistics*, pp. 208-214, 2011.

[22] A. Slivkins, “Contextual Bandits with Similarity Information,” *24th Annual COLT,* 2011.

[23] T. Lu, D. Pal, and M. Pal, “Contextual multi-armed bandits,” *ACM AISTATE*, 2010.

[24] G. Shani, D. Heckerman, and R. I. Brafman, “An MDP-Based Recommender System,” *ACM J. Machine Learning*, vol. 6, pp. 1265-1295, 2005.

[25] S. Pandey, D. Chakrabarti, and D. Agarwal, “Multi-armed bandit problems with dependent arms,” in *Proc. 24th ACM Int. Conf. Machine learning*, pp. 721-728, 2007.

[26] R. Kleinberg, A. Slivkins, and E. Upfal, “Multi-armed bandits in metric spaces,” *in Proc. 40th annual ACM symp. on Theory of Comput.*, 2008.

[27] A. Slivkins, F. Radlinski, and S. Gollapudi, “Ranked bandits in metric spaces: learning diverse rankings over large document collections,” *J. Machine Learning Research*, vol. 14, pp.399-436, 2013.

[28] R. Agarwala, V. Bafna, M. Farach, M. Paterson, and M. Thorup, “On the approximability of numerical taxonomy (fitting distances by tree metrics),” *SIAM Journal on Computing*, vol. 28, no. 3, pp. 1073-1085, 1999.

[29] L. Song, C. Tekin, and M. van der Schaar, “Appendix,” Available: medianetlab.ee.ucla.edu/~linqi/appendix_recommendation.pdf