Adaptive Sequential Recommendation for Discussion Forums on MOOCs using Context Trees

Fei Mi Boi Faltings
Artificial Intelligence Lab
École polytechnique fédérale de Lausanne, Switzerland
firstname.lastname@epfl.ch

ABSTRACT

Massive open online courses (MOOCs) have demonstrated growing popularity and rapid development in recent years. Discussion forums have become crucial components for students and instructors to widely exchange ideas and propagate knowledge. It is important to recommend helpful information from forums to students for the benefit of the learning process. However, students or instructors update discussion forums very often, and the student preferences over forum contents shift rapidly as a MOOC progresses. So, MOOC forum recommendations need to be adaptive to these evolving forum contents and drifting student interests. These frequent changes pose a challenge to most standard recommendation methods as they have difficulty adapting to new and drifting observations. We formalize the discussion forum recommendation problem as a sequence prediction problem. Then we compare different methods, including a new method called context tree (CT), which can be effectively applied to online sequential recommendation tasks. The results show that the CT recommender performs better than other methods for MOOCs forum recommendation task. We analyze the reasons for this and demonstrate that it is because of better adaptation to changes in the domain. This highlights the importance of considering the adaptation aspect when building recommender system with drifting preferences, as well as using machine learning in general.

Keywords

MOOCs forum recommendation, context tree, model adaptation

1. INTRODUCTION

With the increased availability of data, machine learning has become the method of choice for knowledge acquisition in intelligent systems and various applications. However, data and the knowledge derived from it have a timeliness, such that in a dynamic environment not all the knowledge acquired in the past remains valid. Therefore, machine learning models should acquire new knowledge incrementally and adapt to the dynamic environments. Today, many intelligent systems deal with dynamic environments: information on websites, social networks, and applications in commercial markets. In such evolving environments, knowledge needs to adapt to the changes very frequently. Many statistical machine learning techniques interpolate between input data and thus their models can adapt only slowly to new situations. In this paper, we consider the dynamic environments for recommendation task. Drifting user interests and preferences [3, 11] are important in building personal assistance systems, such as recommendation systems for social networks or for news websites where recommendations need be adaptive to drifting trends rather than recommending obsolete or well-known information. We focus on the application of recommending forum contents for massive open online courses (MOOCs) where we found that the adaptation issue is a crucial aspect for providing useful and trendy information to students.

The rapid emergence of some MOOC platforms and many MOOCs provided on them has opened up a new era of education by pushing the boundaries of education to the general public. In this special online classroom setting, sharing with your classmates or asking help from instructors is not as easy as in traditional brick-and-mortar classrooms. So discussion forums there have become one of the most important components for students to widely exchange ideas and to obtain instructors' supplementary information. MOOC forums play the role of social learning media for knowledge propagation with increasing number of students and interactions as a course progresses. Every member in the forum can talk about course content with each other, and the intensive interaction between them supports the knowledge propagation between members of the learning community.

The online discussion forums are usually well structured via the different threads which are created by students or instructors; they can contain several posts and comments within the topic. An example of the discussion forum from a famous "Machine Learning" course by Andre Ng on Coursera[1] is shown in Figure 1. The left figure shows various threads and the right figure illustrates some replies within the last thread ("Having a problem with the Collaborative Filtering Cost"). In general, the replies within a thread are related to the topic of the thread and they can also refer to some other threads for supplementary information, like the link in the second reply. Our goal is to point the students towards useful forum threads through effectively mining forum visit patterns.

Two aspects set forum recommendation system for MOOCs apart from other recommendation scenarios. First, student interests and preferences drift fast during the span of a course, which is influenced by the dynamics in forums and the content of the course; second, the pool of items to be recommended and the items them-

[1] https://www.coursera.org/

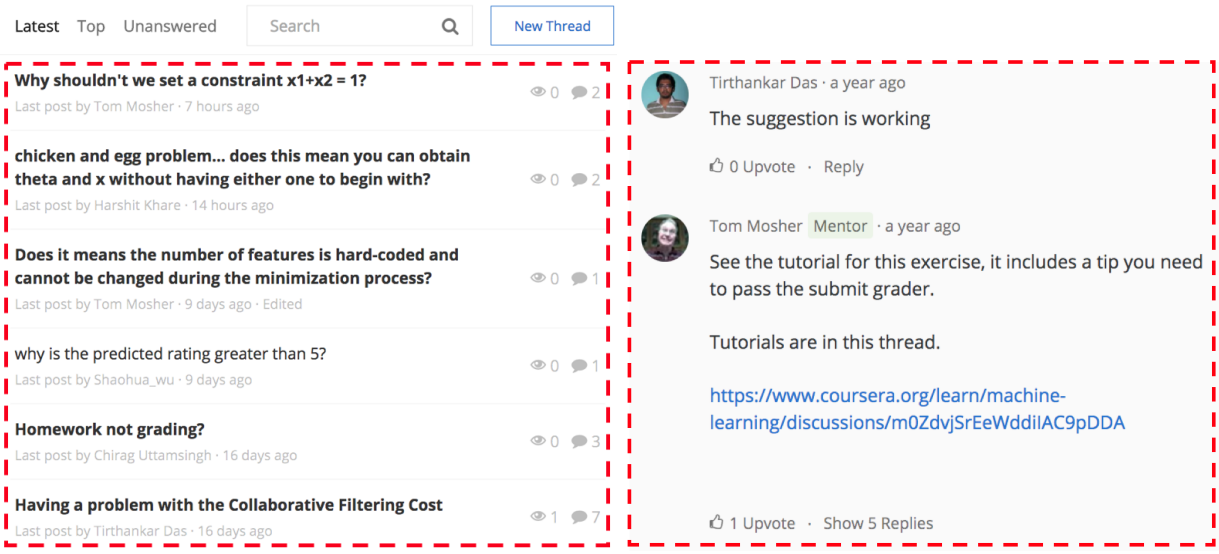

Figure 1: An sample discussion forum. Left: sample threads. Right: replies within the last thread ("Having a problem with the Collaborative Filtering Cost").

selves are evolving over time because forum threads can be edited very frequently by either students or instructors. So the recommendations provided to students need to be adaptive to these drifting preferences and evolving items. Traditional recommendation techniques, such as collaborative filtering and methods based on matrix factorization, only adapt slowly, as they build an increasingly complex model of users and items. Therefore, when a new item is superseded by a newer version or a new preference pattern appears, it takes time for recommendations to adapt. To better address the dynamic nature of recommendation in MOOCs, we model the recommendation problem as a dynamic and sequential machine learning problem for the task of predicting the next item in a sequence of items consumed by a user. During the sequential process, the challenge is combining old knowledge with new knowledge such that both old and new patterns can be identified fast and accurately. We use algorithms for sequential recommendation based on variable-order Markov models. More specifically, we use a structure called context tree (CT) [21] which was originally proposed for lossless data compression. We apply the CT method for recommending discussion forum contents for MOOCs, where adapting to drifting preferences and dynamic items is crucial. In experiments, it is compared with various sequential and non-sequential methods. We show that both old knowledge and new patterns can be captured effectively through context activation using CT, and that this is why it is particularly strong at adapting to drifting user preferences and performs extremely well for MOOC forum recommendation tasks.

The main contribution of this paper is fourfold:

- We applied the context tree structure to a sequential recommendation tasks where dynamic item sets and drifting user preferences are of great concern.
- Analyze how the dynamic changes in user preferences are followed in different recommendation techniques.
- Extensive experiments are conducted for both sequential and non-sequential recommendation settings. Through the experimental analysis, we validate our hypothesis that the CT recommender adapts well to drifting preferences.
- Partial context matching (PCT) technique, built on top of the standard CT method, is proposed and tested to generalize to new sequence patterns, and it further boosts the recommendation performance.

2. RELATED WORK

Typical recommender systems adopt a static view of the recommendation process and treat it as a prediction problem over all historical preference data. From the perspective of generating adaptive recommendations,we contend that it is more appropriate to view the recommendation problem as a sequential decision problem. Next, we mainly review some techniques developed for recommender systems with temporal or sequential considerations.

The most well-known class of recommender system is based on collaborative filtering (CF) [19]. Several attempts have been made to incorporate temporal components into the collaborative filtering setting to model users' drifting preferences over time. A common way to deal with the temporal nature is to give higher weights to events that happened recently. [6, 7, 15] introduced algorithms for item-based CF that compute the time weightings for different items by adding a tailored decay factor according to the user's own purchase behavior. For low dimensional linear factor models, [11] proposed a model called "TimeSVD" to predict movie ratings for Netflix by modeling temporal dynamics, including periodic effects, via matrix factorization. As retraining latent factor models is costly, one alternative is to learn the parameters and update the decision function online for each new observation [1, 16]. [10] applied the online CF method, coupled with an item popularity-aware weighting scheme on missing data, to recommending social web contents with implicit feedbacks.

Markov models are also applied to recommender systems to learn the transition function over items. [24] treated recommendation as a univariate time series problem and described a sequential model with a fixed history. Predictions are made by learning a forest of decision trees, one for each item. When the number of items is big, this approach does not scale. [17] viewed the problem of generating recommendations as a sequential decision problem and they con-

sidered a finite mixture of Markov models with fixed weights. [4] applied Markov models to recommendation tasks using skipping and weighting techniques for modeling long-distance relationships within a sequence. A major drawback of these Markov models is that it is not clear how to choose the order of Markov chain.

Online algorithms for recommendation are also proposed in several literatures. In [18], a Q-learning-based travel recommender is proposed, where trips are ranked using a linear function of several attributes and the weights are updated according to user feedback. A multi-armed bandit model called LinUCB is proposed by [13] for news recommendation to learn the weights of the linear reward function, in which news articles are represented as feature vectors; click-through rates of articles are treated as the payoffs. [20] proposed a similar recommender for music recommendation with rating feedback, called Bayes-UCB, that optimizes the nonlinear reward function using Bayesian inference. [14] used a Markov Decision Process (MDP) to model the sequential user preferences for recommending music playlists. However, the exploration phase of these methods makes them adapt slowly. As user preferences drift fast in many recommendation setting, it is not effective to explore all options before generating useful ones.

Within the context of recommendation for MOOCs, [23] proposed an adaptive feature-based matrix factorization framework for course forum recommendation, and the adaptation is achieved by utilizing only recent features. [22] designed a context-aware matrix factorization model to predict student preferences for forum contents, and the context considered includes only supplementary statistical features about students and forum contents. In this paper, we focus on a class of recommender systems based on a structure, called context tree [21], which was originally used to estimate variable-order Markov models (VMMs) for lossless data compression. Then, [2, 12, 5] applied this structure to various discrete sequence prediction tasks. Recently it was applied to news recommendation by [8, 9]. The most important property of online algorithms is the no-regret property, meaning that the model learned online is eventually as good as the best model that could be learned offline. According to [21], the no-regret property is achieved by context trees for the data compression problem. Regret analysis for CT was conducted through simulation by [5] for stochastically generated hidden Markov models with small state space. They show that CT achieves the no-regret property when the environment is stationary. As we focus on dynamic recommendation environments with time-varying preferences and limited observations, the no-regret property can be hardly achieved while the model adaptation is a bigger issue for better performance.

3. CONTEXT TREE RECOMMENDER

Due to the sequential item consumption process, user preferences can be summarized by the last several items visited. When modeling the process as a fixed-order Markov process [17], it is difficult to select the order. A variable-order Markov model (VMM), like a context tree, alleviates this problem by using a context-dependent order. The context tree is a space efficient structure to keep track of the history in a variable-order Markov chain so that the data structure is built incrementally for sequences that actually occur. A local prediction model, called expert, is assigned to each tree node, it only gives predictions for users who have consumed the sequence of items corresponding to the node. In this section, we first introduce how to use the CT structure and the local prediction model for sequential recommendation. Then, we discuss adaptation properties and the model complexity of the CT recommender.

3.1 The Context Tree Data Structure

In CT, a sequence $\mathbf{s} = \langle n_1, \ldots, n_l \rangle$ is an ordered list of items $n_i \in N$ consumed by a user. The sequence of items viewed until time t is $\mathbf{s}_t$ and the set of all possible sequences $\mathcal{S}$.

A context $S = \{\mathbf{s} \in \mathcal{S} : \xi \prec \mathbf{s}\}$ is the set of all possible sequences in $\mathcal{S}$ ending with the suffix ξ. ξ is the suffix ($\prec$) of $\mathbf{s}$ if last elements of $\mathbf{s}$ are equal to ξ. For example, one suffix ξ of the sequence $\mathbf{s} = \langle n_2, n_3, n_1 \rangle$ is given by $\xi = \langle n_3, n_1 \rangle$.

A context tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and edges $\mathcal{E}$ is a partition tree over all contexts of $\mathcal{S}$. Each node $i \in \mathcal{V}$ in the context tree corresponds to a context S_i. If node i is the ancestor of node j then $S_j \subset S_i$. Initially the context tree $\mathcal{T}$ only contains a root node with the most general context. Every time a new item is consumed, the active leaf node is split into a number of subsets, which then become nodes in the tree. This construction results in a variable-order Markov model. Figure 2 illustrates a simple CT with some sequences over an item set $\langle n_1, n_2, n_3 \rangle$. Each node in the CT corresponds to a context. For instance, the node $\langle n_1 \rangle$ represents the context with all sequences end with item n_1.

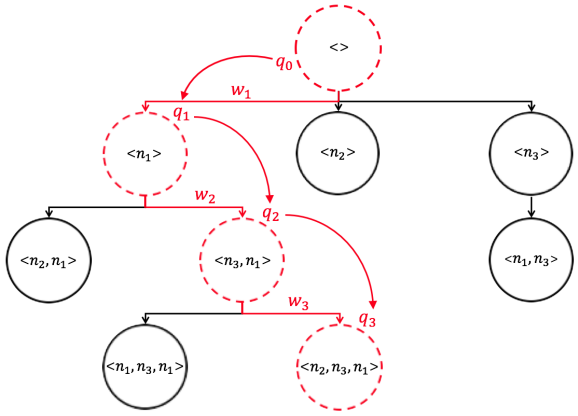

Figure 2: An example context tree. For the sequence $\mathbf{s} = \langle n_2, n_3, n_1 \rangle$, nodes in red-dashed are activated.

3.2 Context Tree for Recommendation

For each context S_i, an expert μ_i is associated in order to compute the estimated probability $\mathbb{P}(n_{t+1}|\mathbf{s}_t)$ of the next item n_{t+1} under this context. A user's browsing history $\mathbf{s}_t$ is matched to the CT and identifies a path of matching nodes (see Figure 2). All the experts associated with these nodes are called *active*. The set of *active* experts $\mathcal{A}(\mathbf{s}_t) = \{\mu_i : \xi_i \prec \boldsymbol{s}_t\}$ is the set of experts μ_i associated to contexts $S_i = \{\mathbf{s} : \xi_i \prec \mathbf{s}_t\}$ such that ξ_i are suffix of $\mathbf{s}_t$. $\mathcal{A}(\mathbf{s}_t)$ is responsible for the prediction for $\mathbf{s}_t$.

3.2.1 *Expert Model*

The standard way for estimating the probability $\mathbb{P}(n_{t+1}|\mathbf{s}_t)$, as proposed by [5], is to use a Dirichlet-multinomial prior for each expert μ_i. The probability of viewing an item x depends on the number of times α_{xt} the item x has been consumed when the expert is active until time t. The corresponding marginal probability is:

$$\mathbb{P}_i(n_{t+1} = x|\mathbf{s}_t) = \frac{\alpha_{xt} + \alpha_0}{\sum_{j \in \mathcal{N}} \alpha_{jt} + \alpha_0} \quad (1)$$

where α_0 is the initial count of the Dirichlet prior

3.2.2 Combining Experts to Prediction

When making recommendation for a sequence $\mathbf{s}_t$, we first identify the set of contexts and active experts that match the sequence. The predictions given by all the active experts are combined by mixing the recommendations given by them:

$$\mathbb{P}(n_{t+1} = x|\mathbf{s}_t) = \sum_{i \in \mathcal{A}(\mathbf{s}_t)} u_i(\mathbf{s}_t)\mathbb{P}_i(n_{t+1} = x|\mathbf{s}_t) \quad (2)$$

The mixture coefficient $u_i(\mathbf{s}_t)$ of expert μ_i is computed in Eq. 3 using the weight $w_i \in [0, 1]$. Weight w_i is the probability that the chosen recommendation stops at node i given that the it can be generated by the first i experts, and it can be updated in using Eq.5.

$$u_i(\mathbf{s}_t) = \begin{cases} w_i \prod_{j:S_j \subset S_i}(1 - w_j), & \text{if } \mathbf{s}_t \in S_i \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

The combined prediction of the first i experts is defined as q_i and it can be computed using the recursion in Eq. 4. The recursive construction that estimates, for each context at a certain depth i, whether it makes better prediction than the combined prediction q_{i-1} from depth $i - 1$.

$$q_i = w_i\mathbb{P}_i(n_{t+1} = x|\mathbf{s}_t) + (1 - w_i)q_{i-1} \quad (4)$$

The weights are updated by taking into account the success of a recommendation. When a user consumes a new item x, we update the weights of the active experts corresponding to the suffix ending before x according to the probability $q_i(x)$ of predicting x sequentially via Bayes' theorem. The weights are updated in closed form in Eq. 5, and a detailed derivation can be found in [5].

$$w_i' = \frac{w_i\mathbb{P}_i(n_{t+1} = x|\mathbf{s}_t)}{q_i(x)} \quad (5)$$

3.2.3 CT Recommender Algorithm

The whole recommendation process first goes through all users' activity sequences over time incrementally to build the CT; the local experts and weights updated using Equations 1 and 5 respectively. As users browse more contents, more contexts and paths are added and updated, thus building a deeper, more complete CT. The recommendation for an activity or context in a sequence is generated using Eq. 2 continuously as experts and weights are updated. At the same time, a pool of candidate items is maintained through a dynamically evolving context tree. As new items are added, new branches are created. At the same time, nodes corresponding to old items are removed as soon as they disappear from the current pool.

The CT recommender is a mixture model. On the one hand, the prediction $\mathbb{P}(n_{t+1} = x|\mathbf{s}_t)$ is a mixture of the predictions given by all the activated experts along the activated path so that it's a mixtures of local experts or a mixture of variable order Markov models whose oder are defined by context depths. On the other hand, one path in a CT can be constructed or updated by multiple users so that it's a mixture of users' preferences.

3.3 Adaptation Analysis

Our hypothesis, which is validated in later experiments, is that the CT recommender can be applied elegantly to domains where adaptation and timeliness are of concern. Two properties of the CT methods are crucial to the goal. First, the model parameter learning process and recommendations generated are online such that the model adapts continuously to a dynamic environment. Second, adaptability can be achieved by the CT structure itself as knowledge is organized and activated by context. New items or paths are recognized in new contexts, whereas old items can still be accessed in their old contexts. It allows the model to make predictions using more complex contexts as more data is acquired so that old and new knowledge can be elegantly combined. For new knowledge or patterns added to an established CT, they can immediately be identified through context matching. This context organization and context matching mechanism help new patterns to be recognized to adapt to changing environments.

3.4 Complexity Analysis

Learning CT uses the recursive update defined in Eq. 4 and recommendations are generated by weighting the experts' predictions along the activated path given by Eq. 2. For trees of depth D, the time complexity of model learning and prediction for a new observation are both $O(D)$. For input sequence of length T, the updating and recommending complexity are $O(M^2)$, where $M = min(D, T)$. Space complexity in the worst case is exponential to the depth of the tree. However, as we do not generate branches unless the sequence occurs in the input, we achieve a much lower bound determined by the total size of the input. So the space complexity is $O(N)$, where N is the total number of observations. Compared with the way that Markov models are learned, in which the whole transition matrix needs to be learned simultaneously, the space efficiency of CT offers us an advantage for model learning. For tasks that involve very long sequences, we can limit the depth D of the CT for space and time efficiency.

4. DATASET AND PROBLEM ANALYSIS

4.1 Dataset Description

In this paper, we work with recommending discussion forum threads to MOOC students. A forum thread can be updated frequently and it contains multiple posts and comments within the topic. As we mentioned before that the challenge is adapting to drifting user preferences and evolving forum threads as a course progresses. For the experiments elaborated in the following section, we use forum viewing data from three courses offered by École polytechnique fédérale de Lausanne on Coursera. These three courses include the first offering of "Digital Signal Processing", the third offering of "Functional Program Design in Scala", and the first offering of "'Reactive Programming'. They are referred to *Course 1*, *Course 2* and *Course 3*. Some discussion forum statistics for the three courses are given in Table 1. From the number of forum participants, forum threads, and thread views, we can see that the course scale increase from *Course 1* to *Course 3*. A student on MOOCs often accesses course forums many times during the span of a MOOC. Each time the threads she views are tracked as one *visit session* by the web browser. The total number of visit sessions and the average session lengths for three courses are presented in Table 1. The length of a session is the number of threads she viewed within a visit session. The thread viewing sequences corresponding to these regular visit sessions are called *separated* sequences in our later experiments and they treat threads in one visit session as one sequence. Models built using separated sequences try to catch short-term patterns within one visit session and we do not differentiate the patterns from different students. Another setting, called *combined* sequences, concatenates all of a student's visit sessions into one longer sequence so that models built using combined sequences try to learn long-term patterns across students. The average length of combined sequences is the average session length times the average number of sessions per student. From *Course 1* to *Course 3*, average lengths for separated and combined sequences both increase.

	Course 1	*Course 2*	*Course 3*
# of forum participants	5,399	12,384	13,914
# of forum threads	1,116	1,646	2,404
# of thread views	130,093	379,456	777,304
# of sessions	19,892	40,764	30,082
avg. session length	6.5	9	25.8
avg. # of sessions per student	3.7	3.3	2.2

Table 1: Course forum statistics for three datasets.

Another important issue that we can discover from the statistics is that thread viewing data available for sequential recommendation is very sparse. For example in *Course 1*, the average session length is 6.5 and the number of threads is around 1116. Then the complete space to be explored will be $1116^{6.5}$, which is much larger than the size of observations (130,093 thread views). The similar data sparsity issue is even more severe in the other two datasets.

4.2 Forum Thread View Pattern

Next, we study the thread viewing pattern which highlights the significance of adaptation issues for thread recommendation. Figure 3 illustrates the distribution of thread views against *freshness* for three courses. The freshness of an item is defined as the relative creation order of all items that have been created so far. For example, when a student views a thread t_m which is the m-th thread created in the currently existing pool of n threads, then *freshness* of t_m is defined as:

$$freshness = \frac{m}{n} \tag{6}$$

We can see from Figure 3 that there is a sharp trend that the new forum threads are viewed much more frequently than the old ones for all three courses. It is mainly due to the fact that fresh threads are closely relevant to the current course progress. Moreover, fresh threads can also supersede the contents in some old ones to be viewed. This tendency to view fresh items leads to drifting user preferences. Such drifting preferences, coupled with the evolving nature of forum contents, requires recommendations adaptive to drifting or recent preferences.

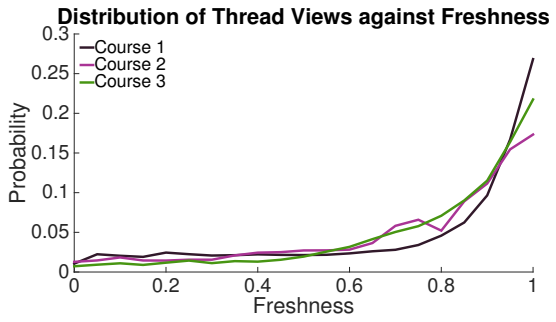

Figure 3: Thread viewing activities against freshness

A further investigation through those views on old threads leads us to a classification of threads into two categories: *general threads* and *specific threads*. Some titles of the general and specific threads are listed in Table 2. We could see the clear difference between these two classes of threads as the general ones corresponds to broad topics and specific ones are related to detailed course contents or exercises. We also found that only a very small part of the old threads are still rather active to be viewed and they are mostly general ones. Different from general threads, specific threads that subject to a fine timeliness are viewed very few times after they get old. In general, sequential patterns are observed more often within specific threads as some specific follow-up threads might be related and useful to the one that you are viewing. So the patterns learned could be used to guide your forum browsing process. On the contrary, sequential patterns on general threads are relatively random and imperceptible.

General Threads	**Specific Threads**
"Using GNU Octave"	"Homework Day 1 / Question 9"
"Any one from INDIA??"	"Quiz for module 4.2"
"Where is everyone from?	"quiz -1 Question 04"
"Numerical Examples in pdf"	'Homework 3, Question 11"
"How to get a certificate"	"Week 1: Q10 GEMA problem"

Table 2: Sample thread titles of general and specific threads.

5. RESULTS AND EVALUATION

In this section, we compare the proposed CT method against various baseline methods in both non-sequential and sequential settings. The results show that the CT recommender performs better than other methods under different setting for all three MOOCs considered. Through the adaptation analysis, we validate our hypothesis that the superior performance of CT recommender comes from the adaptation power to drifting preferences and trendy patterns in the domain. In the end, a regularization technique for CT, called partial context matching (PCT), is introduced. It is demonstrated that PCT helps better generalize among sequence patterns and further boost performance.

5.1 Baseline Methods

5.1.1 Non-sequential Methods

Matrix factorization methods proposed by [23, 22] are the state-of-the-art for MOOCs course content recommendation. Besides the user-based MF given in [23], we also consider item-based MF that generates recommendations based on the similarity of the latent item features learned from standard MF. In our case, each entry in the user-item matrix of MF contains the number of times a student views a thread. We also test a version where the matrix had a 1 for any number of views, but the performance was not as good, so the development of this version was not taken any further. MF models considered here are updated periodically (week-by-week). To enable a fair comparison against non-sequential matrix factorization techniques, we implemented versions where the CT model is updated at fixed time intervals, equal to those of the MF models. In the "One-shot CT" version, we compute the CT recommendations for each user based on the data available at the time of the model update, and the user then receives these same recommendations at every future time step until the next update. This mirrors the conditions of user-based MF. To compare with item-based MF, the "Slow-update CT" version updates the recommendations, but not the model, at each time point based on the sequential forum viewing information available at that time.

5.1.2 Sequential Methods

Sequential methods update model parameters and recommendations continuously as items are consumed. The first two simple methods are based on the observation and heuristic that fresh threads are viewed much frequently than old ones. *Fresh_1* recommends the last 5 *updated* threads, and *Fresh_2* recommends the last 5 *created* threads. Another baseline method, referred as *Popular*, recommends the top 5 threads among the last 100 threads viewed before the current one. We also consider an online version of MF [10] that

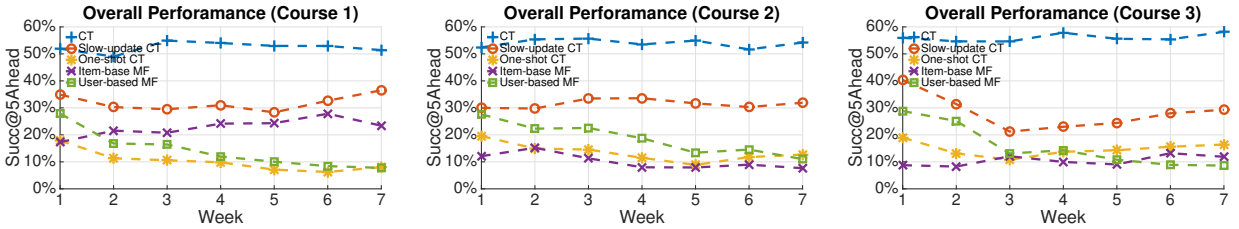

Figure 4: Overall performance comparison of CT and non-sequential methods

is currently the state-of-the-art sequential recommendation method, referred to "online-MF", in which the corresponding latent factor of the item i and user u are updated when a new observation R_{ui} arrive. The model optimization is implemented based on element-wise Alternating Least Squares. The number of latent factors is tuned to be 15, 20, 25 for three datasets, and the regularization parameter is set as 0.01. Moreover, the weight of a new observation is the same as old ones during optimization for achieving the best performance. Furthermore, the proposed CT recommender refers to the full context tree algorithm with a continuously updated model.

5.2 Performance and Adaptation Analysis

5.2.1 Evaluation Metrics

In our case, all methods recommend top-5 threads each time. Two evaluation metrics are adopted in the following experiments:

- **Succ@5**: the mean average precision (MAP) of predicting the immediately next thread view in a sequence.
- **Succ@5Ahead**: the MAP of predicting the future thread views within a sequence. In this case, a recommendation is successful even if it is viewed later in a sequence.

5.2.2 Comparison of Non-sequential Methods

Figure 4 shows the performance comparison between different versions of methods based on MF and CT on three datasets. "CT" is the sequential method with a continuously updated model, and all other methods Figure 4 are non-sequential versions. *Combined* sequences are used for the CT methods here to have a parallel comparison against MF. We found that a small value of the depth limit of the CTs hurts performances, yet a very large depth limit does not increase performance at the cost of computation and memory. Through experiments, we tune depths empirically and set them as 15, 20, 30 for three datasets.

Among non-sequential methods, one-shot CT and user-based MF perform the worst for all three courses, which means that recommending the same content for the next week without any sequence consideration is ineffective. Slow-update CT performs consistently the best among non-sequential methods, and it proves that adapting recommendations through context tree helps boost performance although the model itself is not updated continuously. Compared to slow-update CT, item-based MF performs much worse. They both update model parameters periodically and the recommendations are adjusted given the current observation. However, using the contextual information within a sequence and the corresponding prediction experts of slow-update CT are much more powerful than just using latent item features of item-base MF. Moreover, we can clearly see that the normal CT with continuous update outperforms all other non-sequential methods by a large margin for three datasets. It means that drifting preferences need to be followed though continuous and adaptive model update, so sequential methods are better choices. Next, we focus on sequential methods, and we validate our hypothesis that the CT model has superior performances because it better handles drifting user preferences.

5.2.3 Comparison of Sequential Methods

The results presented in Table 3 show the performance of the full CT recommender compared with other sequential baseline methods under different settings and evaluation metrics. Each result tuple contains the performance on the three datasets. We also consider a *tail performance* metric, referred to *personalized* evaluation, where the most popular threads (20, 30, and 40 for three courses) are excluded from recommendations. The depth limits of CTs using *separated* sequences are set to 8, 10, and 15 for three courses.

We notice that the online-MF method, with continuous model update, performs much worse compared with the CT recommender for all three datasets. This result shows that matrix factorization, which is based on interpolation over the user-item matrix, is not sensitive enough to rapidly drifting preferences with limited observations. The performances of two versions of the *Fresh* recommender are comparable with online-MF, and *Fresh_1* even outperforms online-MF in many cases, especially for **Succ@5Ahead**. It means that simply recommending fresh items even does a better job than online-MF for this recommendation task with drifting preferences. We can see that the CT recommender outperforms all other sequential methods under various settings, except for using non-personalized Succ@5Ahead for *Course 2*. The *Popular* recommender is indeed a very strong contender when using non-personalized evaluation since there is a bias that students can click a "top threads" tag from user interface to view popular threads which are similar to the ones given by *Popular* recommender. From the educational perspective, the setting using separated sequences and personalized evaluation is the most interesting as it reflects shot-term visiting patterns within a session over those specific and less popular forum threads. We could see from the upper right part of Table 3 that the CT recommender outperforms all other methods by a large margin under this setting.

	Non-personalized		*Personalized*	
	Succ@5	Succ@5Ahead	Succ@5	Succ@5Ahead
Separated Sequences				
CT	[**25**, **23**, **21**]%	[**48**, 53, **52**]%	[**19**, **14**, **16**]%	[**41**, **37**, **42**]%
online-MF	[15, 12, 8]%	[33, 29, 23]%	[10, 7 ,6]%	[27, 25, 20]%
Popular	[15, 20, 16]%	[40, **61**, 51]%	[9, 8 ,8]%	[34, 31, 36]%
Fresh_1	[12, 14, 10]%	[37, 43, 41]%	[10, 10, 8]%	[33, 31, 37]%
Fresh_2	[9, 8, 6]%	[31, 31, 29]%	[8, 7, 6]%	[30, 30, 28]%
Combined Sequences				
CT	[**21**, **20**, **20**]%	[**55**, 55, **56**]%	[**16**, **13**, **14**]%	[**46**, **39**, **46**]%
online-MF	[9, 8, 7]%	[34, 27, 23]%	[7,6,6]%	[29, 24, 20]%
Popular	[13, 14, 14]%	[52, **62**, 58]%	[9, 8, 7]%	[45, 36, 43]%
Fresh_1	[10, 12, 9]%	[48, 44, 44]%	[8, 9, 8]%	[44, 34, 42]%
Fresh_2	[7, 6, 6]%	[43, 34, 32]%	[6, 6, 6]%	[42, 32, 31]%

Table 3: Performance comparison of sequential methods

5.2.4 Adaptation Comparison

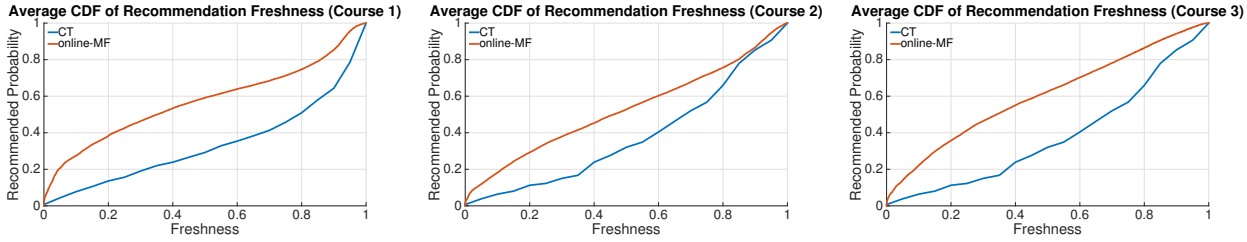

Figure 5: Distribution of recommendation freshness of CT and online-MF

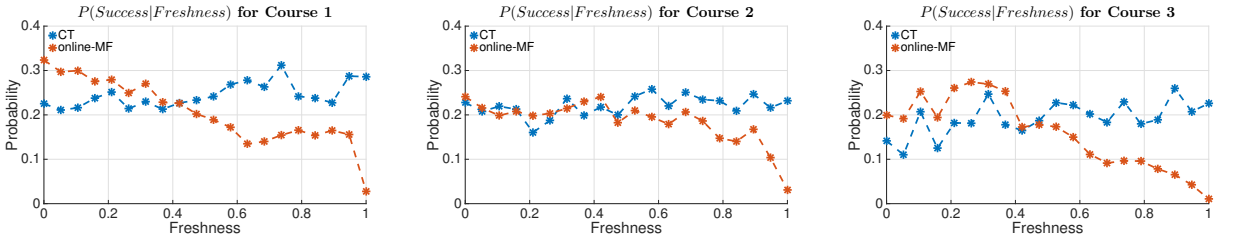

Figure 6: Conditional success rate of CT and online-MF

After seeing the superior performance of the CT recommender, we move to an insight analysis of the results. To be specific, we compare CT and online-MF in terms of their adaptation capabilities to new items. Figure 5 illustrates the cumulative density function (CDF) of the threads recommended by different methods against thread freshness. We can see that the CDFs of CT increase sharply when thread freshness increases, which means that the probability of recommending fresh items is high compared to online-MF. In other words, CT recommends more fresh items than online-MF. As we mentioned before that a large portion of fresh threads are specific ones, instead of general ones, so CT recommends more specific and trendy threads to students while methods based on matrix factorization recommend more popular and general threads.

Other than the quantity of recommending fresh and specific threads, the quality is crucial as well. Figure 6 shows the conditional success rate $P(Success|Freshness)$ across different degrees of freshness for three courses. $P(Success|Freshness)$ is defined as the fraction of the items successfully recommended given the item freshness. For instance, if an item with freshness 0.5 is viewed 100 times throughout a course, then $P(Success|Freshness = 0.5) = 0.25$ means it is among the top 5 recommended items 25 times. As the freshness increases, the conditional success rate of online-MF drops speedily while the CT method keeps a solid and stable performance. It is significant that CT outperforms online-MF by a large margin when freshness is high, in other words, it is particularly strong for recommending fresh items. Fresh items are often not popular in terms of the total number of views at the time point of recommendation. So identifying fresh items accurately implies a strong adaptation power to new and evolving forum visiting patterns. The analysis above validates our hypothesis that the CT recommender can adapt well to drifting user preferences. Another conclusion drawn from Figure 6 is that the performance of CT is as good as online-MF for items with low freshness. This is because that the context organization and context matching mechanism help old items to be identifiable though old contexts. To conclude, CT is flexible at combining old knowledge and new knowledge so that it performances well for items with various freshness, especially for fresh ones with drifting preferences.

5.3 Partial Context Matching (PCT)

At last, we introduce another technique, built on top of the standard CT, to generalize to new sequence patterns and further boost the recommendation performance. The standard CT recommender adopts a complete context matching mechanism to identify active experts for a sequence $\mathbf{s}$. That is, active experts of $\mathbf{s}$ come exactly from the set of suffixes of $\mathbf{s}$. We design a partial context matching (PCT) mechanism where active experts of a sequence are not constrained by exact suffixes, yet they can be those very similar ones. Two reasons bring us to design the PCT mechanism for context tree learning. First, PCT mechanism is a way of adding regularization. Sequential item consumption process does not have to follow exactly the same order, and slightly different sequences are also relevant for both model learning and recommendation generation. Second, the data sparsity issue we discussed before for sequential recommendation setting can be solved to some extent by considering similar contexts for learning model experts. The way PCT does aims to activate more experts to train the model, and to generate recommendations from a mixture of similar contexts.

We will focus on a *skip* operation that we add on top of the standard CT recommender. Some complex operations, like swapping item orders, are also tested, but they do not generate better performance. For a sequence $\langle s_p, \ldots, s_1 \rangle$ with length p, the skip operation generates p candidate partially matched contexts that skip one s_k for $k \in [1 \ldots p]$. All the contexts on the paths from root to partially matched contexts are activated. For example, the path to context $\langle n_2, n_1 \rangle$ can be activated from the context $\langle n_2, n_3, n_1 \rangle$ by the skipping n_3. However, for each partially matched context, there may not exist a fully matched path in the current context tree. In this case, for each partially matched context, we identify the longest path that corresponds it with length q. If q/p is larger than some threshold t, we update experts on this paths and use them to generate recommendations for the current observation. Predictions from multiple paths are combined by averaging the probabilities.

	Success@5	Success@5Ahead	Ratio
PCT-0.5	[+0.4, +0.6, +0.2]%	[+0.8, +0.9, +0.4]%	[4.9, 4.5, 3.3]
PCT-0.6	[+0.5, +0.8, +0.3]%	[+1.1, +1.3, +0.5]%	[4.4, 4.1, 2.9]
PCT-0.7	[+0.7, +0.9, +0.5]%	[+1.6, +1.9, +0.7]%	[3.7, 3.2, 2.5]
PCT-0.8	[+0.8, +1.1, +0.6]%	[+1.9, +2.4, +1.0]%	[3.2, 2.9, 2.1]
PCT-0.9	[+1.0, +1.4, +0.7]%	[+2.0, +2.7, +1.3]%	[2.4, 2.2, 1.4]

Table 4: Performance comparison of PCT against CT for three courses

Table 4 shows the performance of applying PCT for both model update and recommendation with threshold t (PCT-t). Results are compared with the full CT recommender with separated sequences and non-personalized evaluation. For cases where the threshold is smaller than 0.5, we sometimes obtain negative results since partially matched contexts are too short to be relevant. The "Ratio" column is the ratio of the number of updated paths in PCT compared with standard CT. We can see that PCT updates more paths and it offers us consistent performance boosts at the cost of computation.

6. CONCLUSION AND FUTURE WORK

In this paper, we formulate the MOOC forum recommendation problem as a sequential decision problem. Through experimental analysis, both performance boost and adaptation to drifting preferences are achieved using a new method called context tree. Furthermore, a partial context matching mechanism is studied to allow a mixture of different but similar paths. As a future work, exploratory algorithms are interesting to be tried. As exploring all options for all contexts are not feasible, we consider to explore only those top options from similar contexts. Deploying the CT recommender in some MOOCs for online evaluation would be precious to obtain more realistic evaluation.

7. REFERENCES

[1] J. Abernethy, K. Canini, J. Langford, and A. Simma. Online collaborative filtering. *University of California at Berkeley, Tech. Rep*, 2007.

[2] R. Begleiter, R. El-Yaniv, and G. Yona. On prediction using variable order Markov models. *Journal of Artificial Intelligence Research*, pages 385–421, 2004.

[3] R. M. Bell, Y. Koren, and C. Volinsky. The Bellkor 2008 solution to the Netflix prize. *Statistics Research Department at AT&T Research*, 2008.

[4] G. Bonnin, A. Brun, and A. Boyer. A low-order Markov model integrating long-distance histories for collaborative recommender systems. In *International Conference on Intelligent User Interfaces*, pages 57–66. ACM, 2009.

[5] C. Dimitrakakis. Bayesian variable order Markov models. In *International Conference on Artificial Intelligence and Statistics*, pages 161–168, 2010.

[6] Y. Ding and X. Li. Time weight collaborative filtering. In *ACM International Conference on Information and Knowledge Management*, pages 485–492. ACM, 2005.

[7] Y. Ding, X. Li, and M. E. Orlowska. Recency-based collaborative filtering. In *Australasian Database Conference*, pages 99–107. Australian Computer Society, Inc., 2006.

[8] F. Garcin, C. Dimitrakakis, and B. Faltings. Personalized news recommendation with context trees. In *ACM Conference on Recommender Systems*, pages 105–112. ACM, 2013.

[9] F. Garcin, B. Faltings, O. Donatsch, A. Alazzawi, C. Bruttin, and A. Huber. Offline and online evaluation of news recommender systems at swissinfo.ch. In *ACM Conference on Recommender Systems*, pages 169–176. ACM, 2014.

[10] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua. Fast matrix factorization for online recommendation with implicit feedback. In *International ACM Conference on Research and Development in Information Retrieval*, volume 16, 2016.

[11] Y. Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[12] S. S. Kozat, A. C. Singer, and G. C. Zeitler. Universal piecewise linear prediction via context trees. *IEEE Transactions on Signal Processing*, 55(7):3730–3745, 2007.

[13] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *International Conference on World Wide Web*, pages 661–670. ACM, 2010.

[14] E. Liebman, M. Saar-Tsechansky, and P. Stone. DJ-MC: A reinforcement-learning agent for music playlist recommendation. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 591–599. IFAAMAS, 2015.

[15] N. N. Liu, M. Zhao, E. Xiang, and Q. Yang. Online evolutionary collaborative filtering. In *ACM Conference on Recommender Systems*, pages 95–102. ACM, 2010.

[16] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11(Jan):19–60, 2010.

[17] G. Shani, R. I. Brafman, and D. Heckerman. An MDP-based recommender system. In *Conference on Uncertainty in Artificial Intelligence*, pages 453–460. Morgan Kaufmann Publishers Inc., 2002.

[18] A. Srivihok and P. Sukonmanee. E-commerce intelligent agent: personalization travel support agent using Q-Learning. In *International Conference on Electronic Commerce*, pages 287–292. ACM, 2005.

[19] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009:4, 2009.

[20] X. Wang, Y. Wang, D. Hsu, and Y. Wang. Exploration in interactive personalized music recommendation: a reinforcement learning approach. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 11(1):7, 2014.

[21] F. M. Willems, Y. M. Shtarkov, and T. J. Tjalkens. The context-tree weighting method: Basic properties. *IEEE Transactions on Information Theory*, 41(3):653–664, 1995.

[22] D. Yang, D. Adamson, and C. P. Rosé. Question recommendation with constraints for massive open online courses. In *ACM Conference on Recommender Systems*, pages 49–56. ACM, 2014.

[23] D. Yang, M. Piergallini, I. Howley, and C. Rose. Forum thread recommendation for massive open online courses. In *Educational Data Mining*, 2014.

[24] A. Zimdars, D. M. Chickering, and C. Meek. Using temporal data for making recommendations. In *Conference on Uncertainty in Artificial Intelligence*, pages 580–588. Morgan Kaufmann Publishers Inc., 2001.