Learning Context-Dependent Personal Preferences for Adaptive Recommendation

KEITA HIGUCHI, The University of Tokyo, Japan
HIROKI TSUCHIDA, The University of Tokyo, Japan
ESHED OHN-BAR, Carnegie Mellon University, USA and Boston University, USA
YOICHI SATO, The University of Tokyo, Japan
KRIS KITANI, Carnegie Mellon University, USA

We propose two online-learning algorithms for modeling the personal preferences of users of interactive systems. The proposed algorithms leverage user feedback to estimate user behavior and provide personalized adaptive recommendation for supporting context-dependent decision making. We formulate preference modeling as online prediction algorithms over a set of learned policies, *i.e.,* policies generated via supervised learning with interaction and context data collected from previous users. The algorithms then adapt to a target user by learning the policy that best predicts that user's behavior and preferences. We also generalize the proposed algorithms for a more challenging learning case in which they are restricted to a limited number of trained policies at each time step, *i.e.,* for mobile settings with limited resources. While the proposed algorithms are kept general for use in a variety of domains, we developed an image-filter-selection application. We used this application to demonstrate how the proposed algorithms can quickly learn to match the current user's selections. Based on these evaluations, we show that (1) the proposed algorithms exhibit better prediction accuracy compared to traditional supervised learning and bandit algorithms, (2) our algorithms are robust under challenging limited prediction settings in which a smaller number of expert policies is assumed. Finally, we conducted a user study to demonstrate how presenting users with the prediction results of our algorithms significantly improves the efficiency of the overall interaction experience.

CCS Concepts: • **Human-centered computing** → **Human computer interaction (HCI)**; *Ubiquitous and mobile computing*; • **Computing methodologies** → Machine learning approaches.

Additional Key Words and Phrases: Online preference modeling; Context-dependent decision-making

ACM Reference Format:
Keita Higuchi, Hiroki Tsuchida, Eshed Ohn-Bar, Yoichi Sato, and Kris Kitani. 2020. Learning Context-Dependent Personal Preferences for Adaptive Recommendation. *ACM Trans. Interact. Intell. Syst.* 10, 3, Article 23 (November 2020), 27 pages. https://doi.org/10.1145/3359755

1 INTRODUCTION

Context-dependent decision making is a cognitive process of selecting an optimal action from a set of actions depending on the current situational context. In the daily use of interactive systems

Authors' addresses: Keita Higuchi (current affiliation: Preferred Networks), The University of Tokyo, 7-3-1, Hongo,Bunkyo-ku,Tokyo, Tokyo, 1130033, Japan, khiguchi@acm.org; Hiroki Tsuchida, The University of Tokyo, tsuchida@iis.u-tokyo.ac.jp; Eshed Ohn-Bar, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, 15213, USA, Boston University, One Silber Way, Boston, MA, 02215, USA, eshedob@gmail.com; Yoichi Sato, The University of Tokyo, 7-3-1, Hongo,Bunkyo-ku,Tokyo, Tokyo, 1130033, Japan, ysato@iis.u-tokyo.ac.jp; Kris Kitani, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, 15213, USA, kkitani@cs.cmu.edu.

2160-6455/2020/11-ART23 $15.00
https://doi.org/10.1145/3359755

such as smartphones, users often need to perform context-dependent decision making to achieve their goals. For instance, users of a photo-sharing social networking service (SNS) (*e.g.*, Instagram) repeatedly select their photos and add aesthetic image filters then upload them to their accounts. Previous studies reported that repeated decision making increases cognitive load and can cause decision fatigue, which can lead to irrational (sub-optimal) decisions [3, 31].

The overall goal of our research is to build a human-computer interaction (HCI) framework to support repeated context-dependent decision making on the daily use of interactive systems. Especially, we are interested in supporting *user selection* that frequently occurs in diverse contexts to select actions from several candidates based on user preferences. Towards this goal, we aim to develop an automated recommendation method for predicting user actions on the basis of past data and the current situational context. As a concrete example, we develop an image-filter-selection application to predict user selection from a list of image filters on the basis of an estimation of a user's preferences. We expect that such prediction and suggestion will help users select from possible choices (*e.g.*, preset image filters). However, to assist user decision-making of filter selection, the prediction and suggestion should reflect user preferences and context information (*i.e.*, features of given images).

We view this problem as user-preference modeling that adapts user preferences from sequential observations of context information and user selection. This user modeling setting is aimed at building a personalized policy to predict selections of a target user. However, building a personalized policy usually requires a large amount of target-user selections as training data. To reduce the amount of training data, we take a data-driven approach for predicting target-user selections by using a database of other users' selections. With such a database, we can follow machine learning behaviors that compute the features of context information and apply the data to supervised learning algorithms to predict target-user selections.

We propose two online learning algorithms of user modeling to learn personal user preferences for predicting context-dependent user selection on the basis of online prediction approaches [7, 8]. Our learner has a set of policies that are learned with supervised learning from a dataset created from other users. This approach then quickly adapts to target-user preferences by learning and tracking the optimal policies that best describe the target-user selections.

We consider a setting of this online-prediction problem for each of our proposed algorithms. The algorithm for the first setting is allowed to obtain prediction results from all policies in a set. It updates the weights of all the policies sequentially to find an optimal policy for the given target user. We call this online-prediction setting "All Prediction". The algorithm for the second setting is allowed to obtain a limited number of prediction results from the number of policies. We call this setting "Limited Prediction." This algorithm efficiently searches for an optimal policy from such limited observations. This Limited Prediction setting sometimes occurs when an online learner has a large number of policies in a set or computational resources are limited such as on mobile devices.

The main contribution of this work is twofold: (1) we developed two algorithms for online personal preference modeling in both All Prediction and Limited Prediction settings, and (2) applied the proposed algorithms to an image-filter-selection application we developed with a personalized recommendation function, as shown in Figure 1. To evaluate the proposed algorithms, we created a novel dataset for user selections of image filters. We conducted a series of evaluations and a pilot user study to investigate how the filter recommendations from the proposed algorithms help improve the user-selection process. We summarize the findings from our evaluations as follows:

- The proposed algorithm for the All Prediction setting could model personal preferences for each user dataset over baselines. The prediction performance also could be tuned by settings of reward functions.

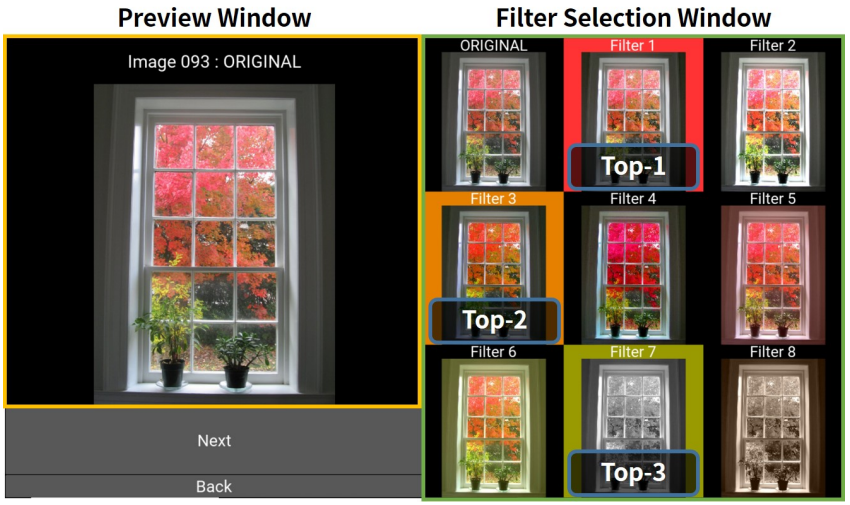

Fig. 1. Image-filter-selection application. Application contains preview window and filter-selection window. Preview window shows original/filtered images that user can check with selected filters. Application shows top-1, -2, and -3 recommended filters, which are highlighted in red, orange, and green, respectively.

- The prediction performance of this algorithm was able to improve as the number of policies in a set increased.
- The proposed algorithm for the Limited Prediction setting could model the personal preferences of user selections using few predictions from the subsets of policies.
- In our pilot study, we observed the potential effectiveness of the recommendation interface of our image-filer-selection application to assist context-dependent user selections.

The rest of the paper is organized as follows. In Section 2, we introduce previous studies on data-driven recommendation and personalization for interactive systems. In Section 3, we describe the problem formulation and our approach as well as the proposed algorithm for the All Prediction setting. In Section 4, we give details on the problem definition of the Limited Prediction setting and the proposed algorithm for this setting. In Sections 5 and 6, we present our image-filter-selection application and data collection for the evaluations, respectively. In Section 7, we present the main results of our evaluations for both problem settings. In Section 8, we describe our pilot user study to investigate the effectiveness of the recommendations from our algorithms on context-dependent user selection of image filters. In Section 9, we provide discussions on our evaluations and user study as well as open problems. Finally, we conclude the paper and mention future work in Section 10.

2 RELATED WORK

2.1 Recommendation based on Contents and User Preferences

Traditional recommendation systems have been developed using collaborative filtering techniques with other user histories [1]. This approach significantly impacts several web services such e-commerce sites [22, 27] and music recommendation [5, 20, 32]. Although they have several computational advantages, collaborative filtering techniques require matrix representation for users and content. They thus cannot be easily applied if all users rate different items. For example, in an

image-filter selection for a photo-sharing SNS, users have individualized content (images) for which to select filters to upload to their accounts. We thus decided to take a learning-based approach that predicts user selections from context information (features) of given content.

Multi-armed bandit is a problem setting on online machine learning in which an online learner aims to find an optimal (most rewarded) arm (policy) from a pre-set of arms based on sequential observations. Previous studies used bandit algorithms for recommendation of news articles [21] and e-commerce [6]. In the bandit settings, an online learner selects and uses an arm from sets then updates the score of the selected arm for balancing between exploitation and exploration. In contrast, we focused on another online learning setting that allow an online learner to observe multiple or all predictions from a set of policies (All Prediction setting).

2.2 Bandit Approaches for Limited Observation Settings

Recent studies considered a novel bandit setting in which an online learner is allowed to access a limited number of arms (more than one and less than the numbers of arms) [28, 34]. Unlike the bandit problem, a learner is allowed to update scores for multiple arms. The studies by Seldin *et al.*and Kale *et al.*proposed algorithms for a limited advice setting in which a learner can access a fixed number M of arms from N arms in each time step ($1 < M < N$) [16, 28]. Yun *et al.*also proposed an algorithm for a similar setting that takes into account observation costs of arms and current budgets, *i.e.*, M can be different in each time step. Qin *et al.*proposed a contextual combination bandit algorithm that a learner selects a set of arms as a super arm then receives user feedback to update scores of selected items [26].

We also consider the limited observation setting with a fixed number of observable policies (Limited Prediction setting) in which our algorithm for this setting effectively finds a combination of policies for personalized recommendation. With such a setting, the previous algorithms use an arm and update scores of $M - 1$ arms in each time step [28, 34]. Unlike the previous algorithms, we focus on a combination of existing user preferences to model a new user preference. To this end, our limited prediction algorithm uses a weighted combination of M policies and updates the M policies in each time step.

2.3 User Preference Modeling for Interactive Systems

HCI studies have introduced principles of preference learning [13] to generate personalized user interfaces [10, 14, 23]. Findlater and Wobbrock proposed a personalized learning method for touchscreen typing [11]. Recent studies proposed preference learning techniques for mining user behaviors [2, 12, 24, 35]. Fogarty *et al.*proposed Cueflik, which uses a concept learning technique to provide a personalized image search by using explicit user feedback [12]. Mehrotra *et al.*proposed PrefMiner, a personalization method for smart notifications on smartphone devices [24]. PrefMiner learns necessity notifications on the basis of mining their interactions with mobile phones. Our algorithms directly learns target user preferences from sequential user selections to assist users' decision making. Our learner then updates a preference model for the target user.

Color enhancement systems for photo images based on user preferences have recently been proposed. Koyama *et al.*proposed an intelligent interface for visual design exploration that introduces general preference analysis using crowd-sourcing [17]. Since this approach explores general preferences for given images, individual preference is sometimes eliminated after a parameter exploration process. In contrast, our approach involves learning personal preferences of target users to predict a target user's selection on the basis of content information and preferences.

Koyama *et al.*also presented the SelPh system, which provides support functions for photo color enhancement by sequentially learning personal preferences in parameter spaces [18]. Learning personal preferences for diverse context spaces (*e.g.*, photos) sometimes requires a large amount

of user feedback. In contrast, our approach involves using other user data to generate multiple individual policies then quickly detecting the optimal policy that best predicts the target user's selection. Note that our aim is to predict user selections from discrete candidates of possible actions.

3 ONLINE PERSONAL PREFERENCE USER MODELING

Our algorithms are used for user modeling that can predict the current user's preferences in an online manner. The algorithms leverage pre-trained policies learned in a supervised manner from previous users for the interactive system to adapt to the current target user. We develop a practical application in which we have access to previously observed user data, which is practical for applications using ubiquitous technologies. Specifically, the proposed algorithms use previous user data of observed contexts (*e.g.*, image features) and user selections (*e.g.*filter selection) as a dataset for learning a set of policies in a supervised manner. Each policy can be used to predict user selection from a feature vector of context information, but this learned policy is assumed to be user dependent. In the online learning phase, we must determine how to leverage this set of policies to best accommodate the user currently interacting with the system. Hence, our approach combines supervised learning *i.e.,* the set of policies, with the All Prediction setting *i.e.,* modeling the preferences of the current user.

Figure 2 shows a conceptual diagram of our image-filter-selection application with our online user modeling. The application contains an interactive system, our online learner, a feature extractor, and set of policies for predicting user selection. Our online learner outputs a prediction result to the interactive system to assist a target user in selecting filters. Our learner receives an image as context information in each time step and extracts context features to obtain a set of prediction results from each policy. The learner also selects a prediction result on the basis of the importance of each policy. The learner also receives user selection as an action label to update the importance in each time step.

We propose online-learning algorithms for two realistic scenarios. We first study an online learning algorithm for the All Prediction setting [7, 8] to find the best performing policy from an existing set of policies. In this setting, we assume unconstrained computing resources so that our algorithm could detect the sequential observation of a target user and detect the best performing policy out of the **entire** set of previously observed policies. Our learner chooses a policy with the current optimal performance in each selection step and maintains a weight over all previously observed policies on the basis of prediction performance.

We also considered a more challenging scenario (common to resource-constrained mobile computing) in which the learner is not able to access all predictions from the complete set of policies, *i.e.,* prediction with our algorithm for the Limited Prediction setting. In such a case, the learner is only allowed to obtain a limited number of predictions in each time step. This setting adds an additional learning challenge of being able to select the most informative policies.

3.1 Formulation

The goal with our online-learning algorithms is to predict the current user's selection of an action $a \in \mathcal{A}$ from a set of candidate actions $\mathcal{A}$. The prediction is made in every time step t by using context (environment) information $\mathbf{x}_t \in \mathcal{X}$ from an interactive system. The algorithms use the current context $\mathbf{x}_t$ to predict a probability distribution over the candidate actions $\mathbf{p}_t \in [0,1]^{|\mathcal{A}|}$ $(\Sigma_i p_t^i = 1)$. The application then displays the results to the interactive system, which returns the user selection a_t. A prediction reward can be computed from a_t and prediction results as $r_t = Reward(a_t, \mathbf{p}_t)$ to update the learner for the next prediction.

In our image-filter-selection problem in which the data are streaming and a pair of an observation and partial label (*e.g.*, which filter was selected) are given at each t. Hence, we take a prediction

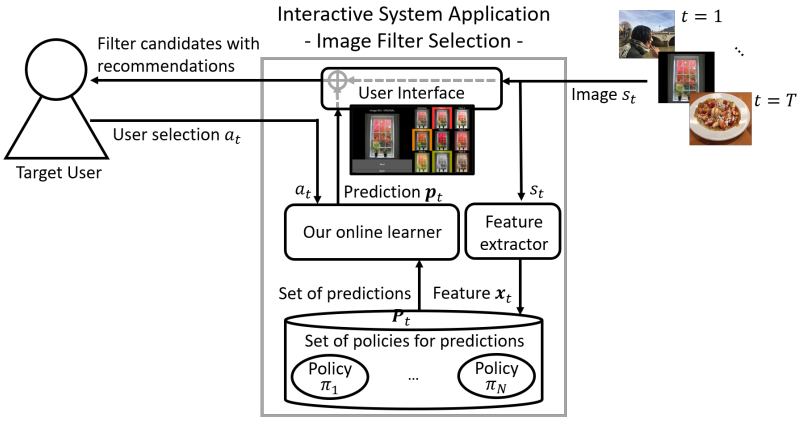

Fig. 2. Conceptual diagram of our image-filter-selection application with our online user modeling

approach with the All Prediction setting to detect the optimal policy for the current target user. In this case, the learner has a set of existing policies $\mathbf{\Pi} = \{\pi_1, \pi_2, ...\pi_N\}$ ($N = |\mathbf{\Pi}|$), where each policy has the ability to predict a user's preferred selections from context information $\pi_i : \mathcal{X} \rightarrow \mathcal{A}$. In a given t, the learning goal is to choose one of the policies to obtain a prediction result $\pi(\mathbf{x}_t) = \mathbf{p_t}$. Consequently, the learner observes the actual a_t.

The success of an online-learning algorithm is measured by its relative cumulative loss on the observed sequence when compared against the loss of a single policy. This value, also known as *regret*, defines the objective function of the learning process. *Regret* is computed by comparing the obtained reward with the reward that would have been obtained by following a single *optimal policy*. Specifically, for each previous example, a set of reward values $\mathbf{R}_t = \{r_{t,1}, ..., r_{t,N}\}$ can be calculated for all policies from the prediction results $\pi_i(\mathbf{x}_t)$ and an a_t. The objective function of the learner is to minimize *regret*

The objective function of the learner is to minimize the *regret* value

$$regret = \max_{\pi \in \mathbf{\Pi}} \sum_{t=1}^{T} Reward(a_t, \pi(\mathbf{x_t})) - \sum_{t=1}^{T} Reward(a_t, \mathbf{p}_t), \tag{1}$$

where the first term is the accumulative reward of the optimal performance policy that has the maximum cumulative reward within the set of policies, and the second term is the total reward of the learner up to the current time. The learning process updates the weights over the policy space to minimize the observed *regret* using

$$w_{t+1,i} = w_{t,i} + r_{t,i} \tag{2}$$

and actions of the agent are determined by choosing the policy with the current highest weight, π_t^*.

We also assume access to a dataset of previously observed N users' data $\mathbf{Z} = \{\zeta_1, ..., \zeta_N\}$. Each ζ (also known as a trajectory) contains a set of the previously observed user selection a and context x information, $\zeta = \{(a_1, x_1), ..., (a_T, x_T)\}$. Using this dataset of previously observed selection-context pairs, we can then learn a policy for predicting each user's selections using supervised learning. This process results in a total of N policies. Given the current context, our proposed online-learning algorithms weigh the N policies to determine the current optimal policy to follow.

ALGORITHM 1: Our algorithm for **All Prediction setting** to predict user selection. Π is set of all estimators. ϕ is feature extractor for s. T is number of time steps. N is total number of policies.

Input: $\mathbf{\Pi} = \{\pi_1, \pi_2, ...\pi_n\}$
$totalReward = 0$
$N = |\mathbf{\Pi}|$
$\mathbf{w} = \{w_i = 0\}$ for $i = 1, ..., N$
for $t = 1, ..., T$ **do**
  $s_t \leftarrow observeStates()$
  $\mathbf{x}_t = \phi(s_t)$
  $\pi_t^* \leftarrow arg\max_{\pi_i \in \Pi} \ w_i$
  $\mathbf{p}_t = \pi_t^*(\mathbf{x}_t)$
  $a_t = GetUserSelection()$
  $totalReward = totalReward + Reward(a_t, \mathbf{p}_t)$
  for $i = 1, ..., N$ **do**
    $\mathbf{p}_i = \pi_i(\mathbf{x}_t)$
    $w_i = w_i + Reward(a_t, \mathbf{p}_i)$
  end
end

In many applications, it is useful to maintain a probability distribution over the candidate actions (*i.e.,* preferred image filters to apply). A list of actions can be presented to the user according to these probabilities in their preferred order.

3.2 Preference Modeling in All Prediction Setting

Algorithm 1 shows the procedure of the proposed algorithm for online user modeling in the All Prediction setting. The algorithm repeatedly predicts probabilities of user selection on the basis of a given context. It also updates weights until a number of time steps T (T can be infinite). The algorithm selects the temporal optimal policy with the maximum weight from a set of all policies. This policy-selection process can also randomly sample a policy if more than two policies have the same maximum weight (*e.g.*, $t = 1$).

This is a general algorithm that can be applied to many types of interactive systems. We thus need to design the following components for making connections between interactive systems and our online learner.

$ObserveState()$ and Feature Extraction $\phi(s_t)$. $ObserveState()$ is the first bridge, which connects interactive systems and our online learner. The learner receives a current state s_t from interactive systems. This can be represented as sensor data and images. The learner then extracts a feature vector $\mathbf{x}_t$ from context information s_t by using a feature extractor ϕ. This is similar to pre-processing for traditional supervised learning.

$GetUserSelection()$. This function is the second bridge, which connects interactive systems and target users. The learner sends a prediction result $\mathbf{p}_t$ to an interactive system. Use of prediction results is the most important design component. We present a concrete interaction scenario and demonstrate a design example of presenting prediction results in Section 5. Users can refer to a form of presentation to make their decision. After user selection, the learner then receives an action label of user selection a_t.

Reward Function $Reward(a_t, \mathbf{p}_t)$. On the basis of a set of prediction results and user selection, our learner computes reward values $Reward(a_t, \mathbf{p}_t)$ to update the cumulative reward and a set of weights $\mathbf{w}$. Designs of reward functions are important because they affect the prediction performance of our learner. We also demonstrate that different designs of reward functions lead to different prediction performances in Section 7.

4 PREFERENCE MODELING IN LIMITED PREDICTION SETTING

We also consider the more challenging setting in which the learner is not able to access all predictions from a set of policies. This setting easily occurs when a set contains a large amount of policies or computing resources are limited (*e.g.*, in mobile computing). In such a case, the learner is allowed to obtain a limited number of predictions in each t. With this setting, the learner aims to prevent prediction accuracy.

4.1 Formulation

Similar to the previous setting, an online learner aims to learn a user preference model and estimate user selection on the basis of predictions of policies from other user data. The main difference from the previous setting is that an online learner is able to access M predictions from a set of N policies (M is constant) in a t. The number of predictions is defined between $1 < M < N$. An online learner can obtain M predictions from any M policies $\mathbf{P}_t = \{\mathbf{p}_1, ..., \mathbf{p}_M\}$.

The online learner aims to learn a preference model for a target user by using a set of policies from other user data. In each t, the online learner predicts probabilities of user selection then obtains actual user selection to update the preference models. We used the same objective function as with the previous setting (Equation 1)

For this problem, we take an approach to learn a combination of M policies from a set for a given target user. Our learner pulls and uses prediction results from M policies then updates the weights for the selected policies with the actual a_t at each t. The learner predicts the probabilities of user selection $\mathbf{q}_t$ by computing the exponentially weighted average of the selected policies' weights as Equation 3

$$\mathbf{q}_t = \sum_{i \in \mathbf{I}_t} \frac{e^{w_i}}{\sum_{j \in \mathbf{I}_t} e^{w_j}} \mathbf{p_i} \tag{3}$$

Let $\mathbf{I}_t \subseteq \{1, ..., N\}$ ($|\mathbf{I}_t| = M$) be the indexes of selected M policies, $\mathbf{w} = \{w_1, ..., w_N\}$ be weights of policies in a set, and $\mathbf{p}_i$ be a prediction result from the $i-$th policy.

Our assumption is that the combination of a small set of multiple policies can effectively model user preference rather than using a policy from a small set. The exponential weighted average over a subset of policies combines predictions and initiatively uses weighted policies. A drawback is that predictions from less weighted policies are used for recommendation when M is not small. To confirm this assumption empirically, We evaluated the performance of our algorithm for the Limited Prediction setting (Section 7.4).

The manner of selecting M policies at each t is important to quickly learn a user preference model for the sequential learning setting. Unlike the first setting, our online learner is not allowed to access all prediction results in a t. To address this problem, we refer to online-learning algorithms for the multi-armed bandit problem in which the problem setting allows an online learner to access an arm in each t. Similar to the All Prediction settings, bandit algorithms find the optimal policy to reduce regret (e.g., Equation 1).

Several bandit algorithms, such as Upper Confidence Bound (UCB) and Kullback-Leibler UCB (KL-UCB) compute scores of arms to choose one in each t. This score is designed to address *"the*

ALGORITHM 2: Our algorithm for **Limited Prediction setting** to predict user selection. Π is set of all estimators, and ϕ is feature extractor for s. T is number of ts, N is total number of policies, and M is number of predictions.

Input: M, $\mathbf{\Pi} = \{\pi_1, \pi_2, ...\pi_n\}$
$totalReward = 0$
$N = |\mathbf{\Pi}|$
$\mathbf{w} = \{w_i = 0\}$ for $i = 1, ..., N$
for $t = 1, ..., T$ **do**
  $s_t \leftarrow observeStates()$
  $\mathbf{x}_t = \phi(s_t)$
  $\mathbf{I}_t \leftarrow SelectGoodPolicyIndices(t)$
  $\mathbf{P}_t = \{\mathbf{p}_i = \pi_i(\mathbf{x}_t)\}$ for $i \in \mathbf{I}_t$
  $\mathbf{q}_t = \sum_{i \in \mathbf{I}_t} \frac{e^{w_i}}{\sum_{j \in \mathbf{I}_t} e^{w_j}} \mathbf{p_i}$
  $a_t = GetUserSelection()$
  $totalReward = totalReward + Reward(a_t, \mathbf{q}_t)$
  for $i \in I_t$ **do**
    $r_i = Reward(a_t, \mathbf{q}_i)$
    $Update(i, r_i)$
    $w_i = w_i + r_i$
  end
end

exploration-exploitation trade-off dilemma," in which the use of current knowledge and search for better arms is balanced. A bandit algorithm usually pulls and uses an arm with the best score in each t.

Our algorithm for the Limited Prediction setting follows the approach of bandit algorithms to choose M policies. Our algorithm computes scores for all policies then uses policies with the top M scores. We carried out implementations of two variants of this algorithm; one with Thompson Sampling algorithm and the other with the KL-UCB algorithm.

4.2 Preference Modeling in Limited Advice Setting

Algorithm 2 shows the proposed algorithm for the Limited Prediction setting. The main differences with Algorithm 1 is that Algorithm 2 obtains indices of M policies $\mathbf{I}_t$ by $SelectGoodPolicy(t)$ and computes exponentially weighted probabilities of user selection $\mathbf{q}_t$. The online learner then updates the scores and weights of the selected policies.

A bandit algorithm is implemented in $SelectGoodPolicy(t)$ and $Update(i, r_i)$ functions. The $SelectGoodPolicy(t)$ function computes all scores of policies based on parameters on the bandit algorithm and returns indices of selected polices in a t. The $Update(i, r_i)$ function receives an index and reward of a policy and updates the parameters for the bandit algorithm.

4.3 Implementation

To investigate efficient arm scores from bandit algorithms, we implemented two variants of our algorithm, one with Thompson Sampling algorithm and the other with the KL-UCB algorithm. Each variant has similar processes for selecting policies but different parameters to compute policy scores.

ALGORITHM 3: Implementation of our limited prediction algorithm with Thompson Sampling algorithm. N is total number of policies, and M is number of predictions. α and β are prior parameters of a Beta distribution, and $Beta(a, b)$ is function to sample value from Beta distribution

Input: N. M. α. β. $\mathbf{A} = \{a_i = 0\}$ for $i = 1, ..., N$. $\mathbf{B} = \{b_i = 0\}$ for $i = 1, ..., N$
Function `SelectGoodPolicies(`t`)`**:**
  $\boldsymbol{\theta} \leftarrow \{Beta(a_i + \alpha, b_i + \beta)\}$ for $i = 1, ..., N$
  $\mathbf{I} \leftarrow argsort(\boldsymbol{\theta})$
  return $\mathbf{I}$ $[N - M : N]$
Function `Update(`i, r`)`**:**
  $a_i = a_i + r$
  $b_i = b_i + 1 - r$

Thompson Sampling Implementation. Thompson Sampling is a heuristic algorithm to address the exploration-exploitation trade-off such as the bandit problem [30]. The concept of Thompson Sampling is to randomly draw an arm on the basis of its probability of being optimal in a t. A recent study conducted empirical evaluations on simulated and real data and found that Thompson Sampling is highly competitive with UCB, which has strong theoretical guarantees on how regret can be proved [9]. An implementation of Thompson Sampling for the Bernoulli bandit algorithm draws a Beta distribution for a score of the i-th arm θ_i as Equation 4.

$$\theta_i \leftarrow Beta(a_i + \alpha, b_i + \beta) \tag{4}$$

Let a_i and b_i be parameters of the i-th arm, which are success and failure counters, respectively. The notations α and β are constant values of prior parameters for a Beta distribution. The Thompson Sampling algorithm selects an arm with the maximum score at a t (see [9] for more details).

Algorithm 3 shows the Thompson Sampling implementation of the proposed limited-prediction algorithm. In the $SelectGoodPolicy(t)$ function, our algorithm draws scores for all policies then returns indices of top M scores. Since we consider a continuous reward value $r \in [0, 1]$, the parameters of the i-th arm a_i and b_i are updated as $a_i + r$ and $b_i + 1 - r$, respectively, in the $Update(i, r)$ function.

KL–UCB Implementation. The KL-UCB algorithm is an online horizon-free index algorithm for stochastic bandit problems [15]. This algorithm uses the Bernoulli KL divergence $KL(p, q) = plog\frac{p}{q} + (1-p)log\frac{1-p}{1-q}$ for $(p, q) \in \Theta^2$ to compute a score for each arm u_i in a t by using Equation 5.

$$u_i \leftarrow \max\{q \in \Phi : c_i KL(\frac{d_i}{c_i}, q) \geq \log(t) + \alpha \log(t)\} \tag{5}$$

Let c_i and d_i be numbers of use and success for the i-th arm, Φ be a value range $[0, 1]$, and α be a constant value. This equation can be computed using Newton iterations for $q \in [0, 1]$ (see [15] for more details).

Algorithm 4 shows the KL-UCB implementation of the proposed limited prediction algorithm. In the $SelectGoodPolicy(t)$ function, our algorithm computes scores for all policies by using Equation 5 then returns indices of top M scores. Since we consider a continuous reward value $r \in [0, 1]$, the parameter d_i of the i-th arm is updated as $d_i + r$ in the $Update(i, r)$ function.

5 APPLICATION OF IMAGE FILTER SELECTION FOR PHOTO-SHARING SERVICE

We applied our user-preference modeling to our image-filter-selection application as a scenario with complexity of context information and user preference. This application imitates photo-sharing

ALGORITHM 4: Implementation of our limited prediction algorithm with KL-UCB algorithm. α is constant value, and $KL(p, q)$ is function for computing KL divergence

Input: N. M. $\mathbf{C} = \{c_i = 0\}$ for $i = 1, ..., N$. $\mathbf{D} = \{d_i = 0\}$ for $i = 1, ..., N$. $\mathbf{\Phi} = [0, 1]$
Function `SelectGoodPolicies(`t`)`**:**
  $\mathbf{U} \leftarrow \{\max\{q \in \Phi : c_i KL(\frac{d_i}{c_i}, q) \geq \log(t) + \alpha \log(t)\}\}$ for $i = 1, ..., N$
  $\mathbf{I} \leftarrow argsort(\mathbf{U})$
  return $\mathbf{I}\ [N - M : N]$
Function `Update(`i, r`)`**:**
  $c_i += 1$
  $d_i += r$

SNSs (*e.g.*, Instagram) with which users select a specific image filter for a given photo. We believe user selections change depending on the content of photos and personal preference. We also believe there is no clear goal and it is difficult to define the best filter for everyone. However, this scenario is very common for many users of photo-sharing SNSs.

We are interested in how accurately the proposed algorithms are able to predict user selections and how filter recommendation affects user selection and behavior. We thus built a prototype interface as a GUI of our image-filter-selection application. The interface features a special function of filter suggestion powered by our online user-preference modeling. With this interface, users are asked to repeatedly select a filter from predefined filters for a given image.

User Interface. Figure 1 shows our GUI interface for image-filter selection. The interface contains several GUI components, i.e., image preview, filter view with the top-3 recommendations, and image sequence buttons (Next and Back). The interface has the original image and eight image filters on the candidate filter view. Users can select a filter by clicking on it with a mouse. The image preview then shows an image with the selected filter applied. Users can freely compare filters to make a selection.

Presentation of Selection Prediction as Suggestions. The interface can highlight three filters with top-3 probability on the filter view, as shown in Figure 1. We make a ranking list of filters on the basis of probability values of prediction results $\mathbf{p}_t$ in each t. We assign red, orange, and green for the top-1, -2, and -3 filters, respectively. Users can see highlighted filters as suggestions based on target-user preference.

Use of Other Users' Filter Selection Histories. The application uses selection histories of other (previous) users that are pairs of images and selected filters. The application has a feature extractor to generate feature vectors of images. It also uses supervised learning algorithms to generate a policy from selection histories of a user and has multiple policies in a set. We describe implementation details in Section 7.

6 DATA COLLECTION

We created a dataset of image-filter selections for our algorithm evaluations and a user study for the image-filter-selection application. We recruited 16 university students (Female: 3, and Male: 13; average age was 25 (SD: 2.55)). Seven of the participants had never selected image filters for photo-sharing SNSs. Due to the use of supervised learning for policy generation, a small amount of selection data (training data) makes policies with poor performance. We thus decided to collect 300 selections from each participant to learn a policy that can reflect the participant's preference. We

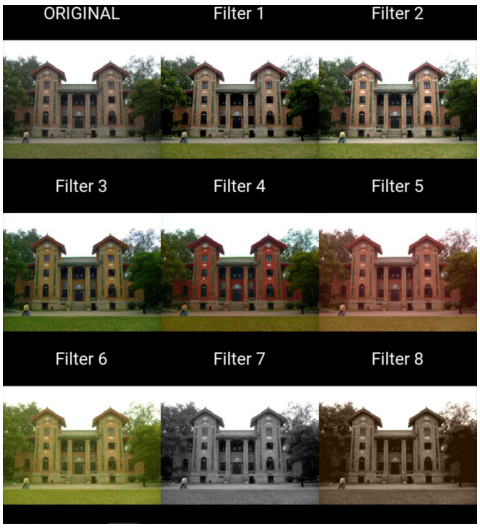

Fig. 3. Our user interface contains several GUI components, i.e., image preview, filter view with top-3 suggestions, and image sequence buttons (Next and Back). Interface displays original image (no filter) and 8 image filters.

believe that 300 selections for image filters is a practical number for active users of actual SNSs (e.g., Instagram).

The participants followed our data-collection process for the dataset. First, they filled out a questionnaire with information such as age, gender, and experience of image-filter selection. They then started selecting image filters for 300 images for which 9 candidates were shown: the original image and 8 filtered images. We asked them to select the candidate for a given image to upload the (original or filtered) images to their accounts on photo-sharing SNSs. During the data collection, participants were allowed to rest anytime if they felt tired. Finally, we obtained 300 user selections of image filters from all 16 participants (4800 pairs of images and filters) for the dataset. Note that the application did not show any recommendations during the data-collection process.

We used the Open Image Dataset[1], which is a large-scale image dataset that contains over ten million images for computer vision competitions. We selected this dataset to preserve diverse and adequate images for the data collection. We thus randomly sampled 300 images from the dataset for each participant, *i.e.*, each participant selected filters for 300 different images.

We designed a set of eight original image filters, as shown in Figure 3. We used nConvert[2], which is a command-line batch-image processor to create the eight image filters. To reproduce the filters, we show command-line parameters in Table 1. We designed the image filters to be common and useful to beautify pictures on the basis of professional advice. Users can freely compare filters to select the original image or a filtered image as an action. An action label (user selection) a_t is assigned, as shown in the second column in Table 1.

Figure 4 shows the filter-selection probabilities of the 16 participants from the dataset. Although ten participants frequently selected filter label 1, there were several types of preference distributions among the participant. The average total selection time of each participant was 64.1 minutes (SD: 25.3) including rests.

[1]Open Images Dataset: https://storage.googleapis.com/openimages/web/index.html
[2]NConvert: Command Line Batch utility for images https://www.xnview.com/en/nconvert/

Table 1. Filter action labels and parameters to reproduce our eight filters with the nConvert software [2] for our dataset

	Original	Filter1	Filter2	Filter3	Filter4	Filter5	Filter6	Filter7	Filter8
action label a_t	0	1	2	3	4	5	6	7	8
Filter params.									
Brightness	–	-10	40	–	–	–	–	10	-25
Contrast	–	25	45	Auto	Auto	–	–	25	25
Gamma	–	1.35	0.8	–	–	–	–	1.1	1.1
Gamma Sat.	–	–	–	2.25	3.0	1.3	1.3	–	–
Hue Rotation	–	–	–	4	-4	–	–	–	–
Color temp.	–	–	–	–	–	35,0,0	35,35,0	–	–
Grayscale	–	–	–	–	–	–	–	Yes	–
Sepia	–	–	–	–	–	–	–	–	Yes

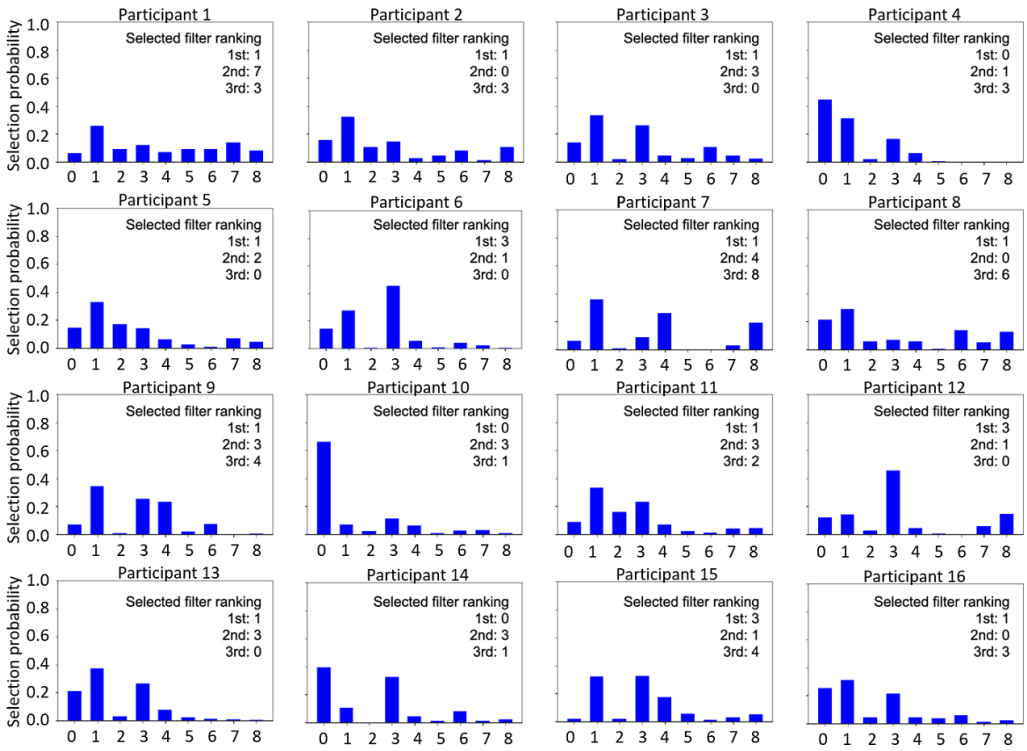

Fig. 4. Selection distributions of 16 participants for our dataset

7 ALGORITHM EVALUATIONS

Using the above dataset, we conducted three evaluations for the proposed algorithms. We 1) compared our all-prediction algorithm with traditional supervised learning and bandit algorithms, 2) investigated the impact of the number of policies in a set, and 3) compared our limited-prediction algorithm with state-of-the-art algorithms.

7.1 Evaluation Settings

We designed our evaluation settings to apply the proposed algorithms to our image-filter-selection application. We decided a way for generating a set of policies and a reward function to sequentially update the proposed algorithms.

7.1.1 Context Features $\mathbf{x}_t$*.* The performance of the proposed algorithms is significantly affected by the performance of supervised learning algorithms that learn existing users' preferences from selection data to generate a set of policies. Therefore, the choice of context features $\mathbf{x}_t$, and supervised learning algorithms were important for our evaluations. In computer vision, recent developments in deep learning network (DNN) techniques outperformed other hand-crafted image features regarding prediction [19]. Yosinski *et al.*showed that middle-layers of trained convolutional NNs can be used to extract feature vectors of images for learning other datasets [33]. In artwork recommendation, Messina *et al.*, revealed that image features from DNN embedding resulted in state-of-the-art prediction performance compared to hand-crafted image features (*e.g.*, color histogram) [25]. Therefore, we selected DNN embedding to extract context features from images. More specifically, we used the ILSVRC VGG-16 FC7 layer [29] as 4096-dimensional deep learning features. We then compressed the 4096-dimensional features to 256-dimensional features using principal component analysis for all image data. We used the Random Forest classifier to generate policies for all evaluation conditions in the three evaluations.

7.1.2 Reward Function $Reward(a^t, \mathbf{p}_t)$*.* The proposed algorithms use a reward function to sequentially update an online learner in each t to model target user preferences. The reward function should be properly designed for a specific application. In this evaluation, we focused on our image-filter-selection application in which the interface visualizes the top-3 recommended filters for a given image. In this case, the top-3 recommendations are expected to contain the actually selected filter. Therefore, we used the reward function that can respectively provide rewards to accurately predicted policies and policies in which the top-3 predictions contain the correct filter.

Algorithm 5 shows the reward function for our image-filter-selection application. This function has two predefined reward parameters for top-1 $R_{top1} = 1$ and top-3 $R_{top3} \in [0, 1)$ accuracy ($R_{top1} > R_{top3}$). In each t, accurately predicted policies receive R_{top1}, and policies in which the top-3 predictions contain the correct filter receive R_{top3}. Otherwise, the reward function gives other policies no reward, i.e., 0. In Evaluation 1, we evaluated the trade-off of top-1 and top-3 accuracies by changing the R_{top3} parameter. Although we considered top-3 accuracy for the reward function, the participants did not see any recommendations in their filter selections.

7.1.3 Evaluation Measures. In our evaluations, we measured top-1 and top-3 prediction accuracies (chance rate: 0.11 and 0.33, respectively) of filter selection. We also measured the mean reciprocal rank (MRR), which is a common metric for evaluating recommendation systems (higher is better). In Evaluation 1, we also compared regret values (Equation 1: lower is better) under conditions that show the difference in reward values with optimal policies. In Evaluation 3 for the Limited Prediction setting, we determined how a limited number of predictions affect processing times.

ALGORITHM 5: Reward Function for Image Filter Selection

Input: Reward values for top-1 $R_{top1} \in [0, 1]$ and top-3 $R_{top3} \in [0, 1)$ accuracy as predefined environmental parameters ($R_{top1} > R_{top3}$)

Function `Reward`(a_t, **p**):

```
ranking = argsort_descending_order(p)
if a_t = ranking_1 then
    return R_top1
else if a_t = ranking_2 or a_t = ranking_3 then
    return R_top3
return 0
```

7.2 Evaluation 1: Prediction Performance in All Prediction Setting

We compared the proposed algorithm for the All Prediction setting with traditional supervised learning and bandit algorithms. The main purposes of this evaluation was evaluating 1) whether the proposed all-prediction algorithm can achieve better prediction performance over baselines, and 2) the effect of the difference of the reward value R_{top3} to determine the tradeoff of top-1 and top-3 accuracies. We conducted this evaluation under a variety of R_{top3} that changed from 0 to 1 in steps of 0.01.

We conducted 16-fold cross-validation, i.e., 1 selection set for test data and the remaining 15 selection sets for train data. We sequentially changed and evaluated each test dataset for all of conditions. Since each test dataset has 300 filter selections, we tested the algorithms with $T = 300$. We compared the following conditions and an optimal condition:

Proposed: Our method for the All Prediction setting. We applied each user selection dataset (15 users) to the Random Forest classifier to generate a set of policies. Our online learner is allowed to access all prediction results in the set and update all policy weights in each t.

Baseline 1: bandit approach UCB. We implemented the UCB algorithm as a bandit baseline [4]. The reason we chose this is that it is deterministic and has a theoretically guaranteed regret bound. Although there are several variations of the UCB algorithm (e.g., KL-UCB), it is easy to understand the scheme for selecting arms. The main difference between the proposed all-prediction algorithm and the bandit UCB algorithm is the number of observations and updates for policies. The UCB algorithm is allowed to access and update only one policy in each t. The UCB algorithm has the same set of policies as the proposed algorithm.

Baseline 2: all-in-one policy π^{all} (traditional supervised learning). We merged all user selection datasets (15 users) into an all-in-one training dataset (4500 pairs). We also applied the data to the supervised learning algorithm to make an all-in-one policy. Based on the findings from our preliminary study, we used the Random Forest classifier to learn the all-in-one policy (the same as with the proposed all-prediction algorithm). Note that this baseline does not change the prediction results if the reward function changes.

Optimal policies π^*. For regret analysis, we computed the optimal policy for each test dataset. We built 15 policies, as with the proposed all-prediction algorithm. We chose the optimal policy with the maximum rewards from 15 policies. Note that the optimal policy can be revealed after T time steps.

Table 2. Results of prediction performance for our image-filter-selection application: we measured top-1 accuracy, top-3 accuracy, MRR, and average regret values.

	Baseline 1 UCB	Baseline 2 π^{all}	Proposed	Optimal π^*
Reward parameter $R_{top3} = 0.0$				
Top-1 Accuracy	0.31 (SD:0.07)	0.23 (SD:0.11)	**0.37 (SD:0.09)**	0.39 (SD:0.08)
Top-3 Accuracy	0.62 (SD:0.09)	**0.71 (SD:0.13)**	0.64 (SD:0.11)	0.66 (SD:0.12)
MRR	0.51 (SD:0.06)	0.48 (SD:0.07)	**0.55 (SD:0.07)**	0.57 (SD:0.08)
Average Regret	24.25 (SD:13.20)	48.25 (SD:45.33)	**5.69 (SD:3.82)**	–
Reward parameter $R_{top3} = 0.5$				
Top-1 Accuracy	0.28 (SD:0.05)	0.23 (SD:0.11)	**0.35 (SD:0.09)**	0.36 (SD:0.09)
Top-3 Accuracy	0.65 (SD:0.10)	0.71 (SD:0.13)	**0.73 (SD:0.12)**	0.74 (SD:0.11)
MRR	0.50 (SD:0.05)	0.48 (SD:0.07)	**0.56 (SD:0.07)**	0.57 (SD:0.07)
Average Regret	26.56 (SD:9.94)	25.63 (SD:20.64)	**4.13 (SD:2.34)**	–

Results of Evaluation 1

Table 2 shows the results of Evaluation 1. The table also includes results of prediction performance on different reward values $R_{top3} = 0, 0.5$. The table shows top-1 accuracy, top-3 accuracy, MRR, as well as regret values for optimal policy π^*. For top-1 accuracy and MRR, the proposed all-prediction algorithm outperformed the baselines. In top-3 accuracy, although baseline 2 (Supervised) provided better results with $R_{top3} = 0$, the proposed all-prediction algorithm outperformed the others with $R_{top3} = 0.5$. These results indicate that our algorithm is able to change prediction performance by designing reward functions. The proposed all-prediction algorithm also had smaller regrets than the baselines.

Figure 5 (A) show more detailed results of prediction performance with top-3 reward parameters R_{top3} in steps of 0.01. The results indicate that the tradeoff of $R_{top3} \in [0, 1)$ affected top-1 and top-3 accuracies. The proposed all-prediction algorithm outperformed baseline 2 (Supervised) when the top-3 reward value was higher than 0.28. In contrast, our algorithm always performed best in terms of top-1 accuracy and MRR. The results of MRR were stable under various reward parameters R_{top3}.

As shown in the left graphs of 5 (B) and (C), the proposed all-prediction algorithm quickly found and tracked optimal policies when the reward parameters were 0.0 or 0.5. In contrast, the UCB algorithm as the bandit baseline increased regret values over time. The center and right graphs of 5 (B) and (C) show that time changes in top-1 accuracy and MRR.

7.3 Evaluation 2: Impact of the Number of Policies in Set

We investigated the impact of increasing the number of policies in a set. We assumed that the proposed all-prediction algorithm can increase prediction performance along with increasing the number of policies. To this end, we evaluated the proposed all-prediction algorithm by changing the number of policies in a set. We conducted k-fold cross-validations ($2 \leq k \leq 16$) for all possible combinations of policies. We selected k user selection data from a dataset then conducted k-fold cross-validation with the selected k data. We applied this to all possible combinations ${}_{16}C_k$. We determined the averages of results from combinations for each k-fold cross-validation. We compared

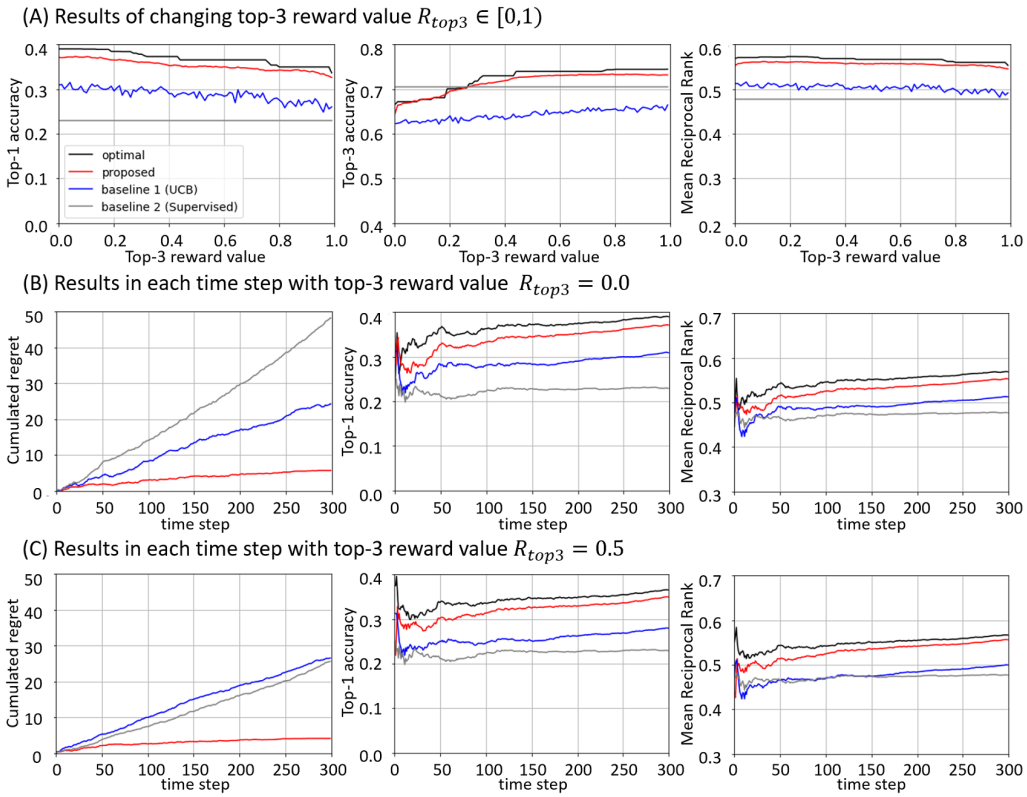

Fig. 5. (A) Results of regret and prediction performance with top-3 reward $R_{top3} \in [0, 1)$. (B) and (C) Evaluation results in each t when reward values were 0.0 and 0.5, respectively.

the proposed all-prediction algorithm with the UCB algorithm as a baseline bandit algorithm. We evaluated two rewards values; $R_{top3} = 0, 0.5$.

Results of Evaluation 2

Figure 6 shows the results of Evaluation 2. The graphs show that the results of the optimal policy and the proposed all-prediction algorithm increased along with the increase in the number of policies in most cases. Our algorithm could track the optimal policy, even when the number of policies increased. Although the results of top-3 accuracy with R_{top3} did not increase, this setting only took into account top-1 accuracy in the reward function. The results of the UCB algorithm did not increase along with the number of policies.

7.4 Evaluation 3: Prediction Performance in Limited Prediction Setting

We evaluated the other proposed algorithm with two variations for the Limited Prediction setting. We assumed that our algorithm is still able to exhibit reasonable prediction accuracy unless it cannot access all the predictions from a set of policies in a t. In this evaluation, we conducted16-fold cross-validation, as in Evaluation 1, but changed the number of predictions M from 1 to 15. We used two reward values $R_{top3} = 0.0$ and $R_{top3} = 0.5$. In addition to the performance metrics, we

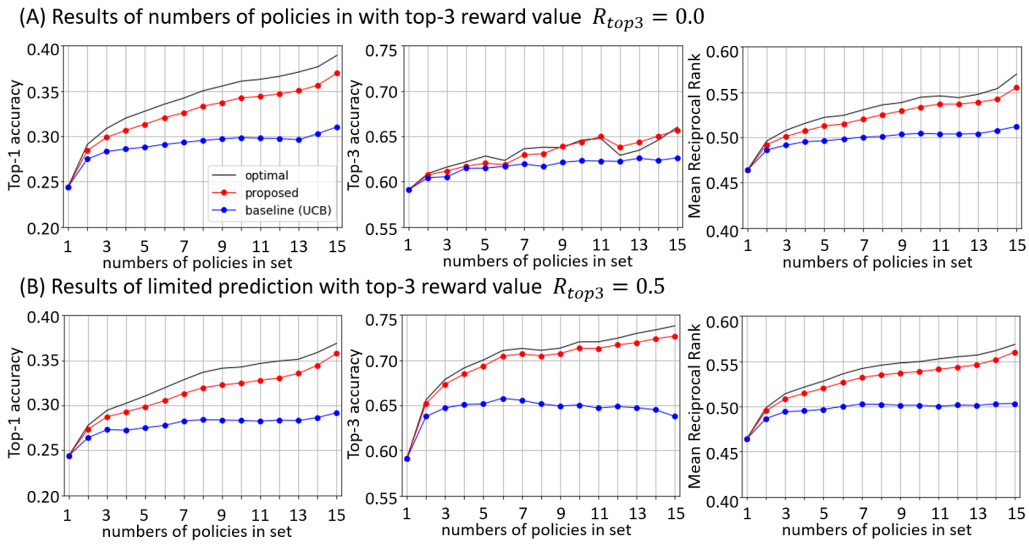

Fig. 6. Evaluation 1 results: we confirmed that the results of optimal policy and proposed all-prediction algorithm increase along with increasing number of policies in set

also measured processing times under different M predictions. For completeness, we also report on the run-time for each algorithm on an Intel Core i7 6700K with 32 GB memory.

We compared our limited-prediction algorithm with state-of-the-art algorithms for similar settings called multi-armed bandit algorithms with additional observation [16, 28, 34]. The main difference between our algorithm and these competitive algorithms is that our algorithm uses an exponentially weighted average of M predictions, but the competitive algorithms use an arm and update scores of $M - 1$ arms. We compared the algorithms under the following conditions (note that all conditions had the same set of 15 policies):

Proposed 1 (Thompson Sampling): Our method with the Thompson Sampling implementation for limited advice setting. We applied the Thompson Sampling algorithm (Algorithm 3) to our algorithm (Algorithm 2). Our online learner is allowed to select M predictions in the set of policies based on bandit scores and update weights and bandit scores of the selected policies in each t.

Proposed 2 (KL-UCB): Our method with the KL-UCB implementation for limited advice setting. We applied the KL-UCB algorithm (Algorithm 4) to our original algorithm (Algorithm 2). Our online learner selects M prediction results from the set of policies and updates the weights and bandit scores of the selected policies.

Competitive method 1 [28] (Limited Advice): multi-armed bandit with limited advice. We implemented an algorithm that selects an arm based on estimated loss values of policies and uses its prediction result [28]. The algorithm also samples $M - 1$ additional policies uniformly without replacement then updates loss values to the selected M policies (See more details of this algorithm in [28]).

Competitive method 2 [34] (KL-UCB-AO): modified version of multi-armed bandit with additional observations. The original multi-arm bandit algorithm is able to change the number of additional

observations for updating bandit scores of arms based on observation costs and budgets, *i.e.*, M can be different at each t. To fit to our setting, we modified this algorithm to take constant M policies at a t. The modified algorithm selects a policy with the highest KL-UCB scores from a set and uses its prediction results. The algorithm also selects additional $M-1$ policies based on KL-UCB scores then updates the KL-UCB scores of the selected M policies.

Results of Evaluation 3

Figures 7 (A) and (B) show the results in the Limited Prediction setting. Overall, our algorithm with the KL-UCB algorithm exhibited better top-1 accuracy than the competitive algorithms when there are few limited predictions M (especially $1 < M < 6$). We empirically confirmed that few combinations of M policies could learn personal preference models over using a policy. When M is close to N, the performance of the proposed limited-prediction algorithm was similar to that of the competitive algorithms in the dataset.

Figure 7 shows the results of processing times for all algorithms. The left graph shows the processing times of policy selection without prediction times of M policies. The proposed limited-prediction algorithm with KL-UCB algorithm and competitive method 2 (KL-UCB-AO) were slower than competitive method 1 and our algorithm with the Thompson Sampling algorithm because each computation of a KL-UCB score requires solving an optimization problem. The right graph shows the overall processing times for all algorithms. Due to the various times of predictions from policies, we added M times of the fixed average prediction time (9.35 ms) to the policy selection time. We confirmed that the prediction times of each policy affected to overall processing times rather than selecting M policies from a set of N policies.

8 USER STUDY

We then conducted a pilot user study in which participants selected image filters for given images using our interface with visualized recommendations. To the best of our knowledge, this study is the first to investigate how visualizing top-N recommendation affects user-selection behavior in image-filter applications. Therefore, the main purpose of this user study was to understand the key effects of visualization. More specifically, we investigated how visualizing recommendation affects participants' filter selection compared with no recommendation (Task 1), and whether the recommendation could reflect participants' personal preference (Task 2). We expected that the recommendation based on the proposed algorithms reflects target user preferences and helps in selecting image filters.

We recruited 16 university students as participants of our user study (Female: 5, and Male: 11; average age 24.1 (SD: 0.96)). They had diverse experiences with image-based SNSs (9 participants had experience of uploading filtered images to their SNS accounts).

We used the proposed algorithm for the All Prediction setting to ensure reasonable prediction accuracy for presenting recommendations. We also used selection data and filter sets of our dataset to use various images from the Open Image Dataset[1]. Regarding the reward function, we fixed reward values to $R_{top1} = 1$ and $R_{top3} = 0$.

8.1 Tasks and Procedure

Task 1. We designed two tasks for this user study. The first task was designed to determine the effectiveness of the top-3 filter recommendations from our all-prediction algorithm. Each participant selected image filters for 100 given images that were sampled from the Open Image Dataset[2]. We investigated under two conditions, one with our all-prediction algorithm with recommendation (Proposed condition) and one without recommendation (None condition). With the None condition, participants did not receive any recommendations from our algorithm. As mentioned above, this is

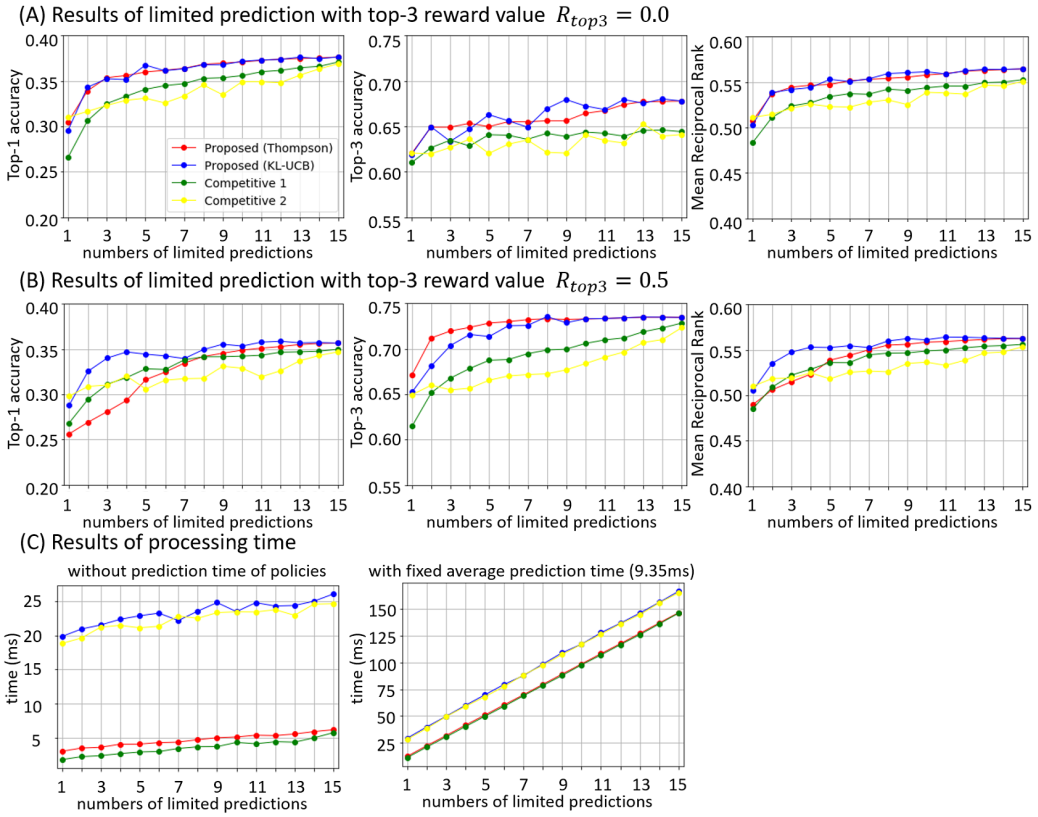

Fig. 7. Results of Evaluation 3 in Limited Prediction setting. Our limited-prediction algorithm with KL-UCB algorithm performed better than competitive methods when M limited predictions M was small. We also confirmed that prediction times of each policies affected the overall processing times rather than selecting M policies.

the first study to determine whether visualizing recommendations helps in filter selection. We thus thought the None condition would be an appropriate baseline rather than other recommendation conditions (e.g., bandit-based recommendation).

Before the first task, participants used both conditions to select filters for less than ten images. We asked the participants to select a suitable filter from nine candidates for a given image to upload the (original or filtered) images to their accounts of photo sharing SNSs. During the task, participants selected 50 image filters with the Proposed condition, and remaining 50 images for the None condition. To avoid effect of learning, the Proposed and None conditions were shown alternatively for each 100 filter selections. Our all-prediction algorithm sequentially updated the weights of policies based on user selections in both None and Proposed conditions.

Task 2. We designed the second task to determine whether the proposed all-prediction algorithm can reflect target-user preferences of filter selections compared to random suggestions. After Task 1, participants worked on task 2 with updated weights of policies (*i.e.*, we used learned models from Task 1). We showed two highlighted filters in red (top-1 recommendation) for given images on the

GUI interface. One of the recommended filters was from the top-1 probability of a prediction result, and the other was randomly selected from remaining candidates. We asked participants to select a candidate from the two recommended filters based on their preference for 50 images. After task 2, participants then filled out a questionnaire.

8.2 Evaluation Measures

We used several quantitative metrics for the Proposed and None conditions. In task 1, we used median selection times and median number of filter comparisons for 50 images under each condition. We also measured reference and selection ratios of the top-1 recommendation to determine whether participants used the recommended information for their selection. The reference ratio is the percentage of participants who clicked the (referred) top-1 recommendation at least once. We also determined the first-selection ratio of the top-1 recommendation, which is the percentage of whether participants started to review the top-1 recommendation. Since the interface initially shows original images, we did not change reference ratio when the original images had the top-1 recommendation. The selection ratio is the percentage of participants who selected the top-1 recommendation. We computed filter selection histograms to determine the similarities of selection distributions under both conditions. We measured the similarities of histograms from each condition based on histogram intersection. In task 2, we also determined the selection ratios of highlighted filters with top-1 probability of prediction results and random selection to determine whether the proposed all-prediction algorithm was able to reflect participants' preference. We also conducted the Wilcoxon signed-rank test to determine whether there were significant differences. We adjusted the results of p-values using the Benjamini–Hochberg procedure for multiple comparisons.

We also observed participants' experiences through qualitative feedback after their tasks. Specifically, we had participants respond to the following items: Q1) "I felt that the recommended filters reflected my preference gradually." Q2) "I felt that the recommended filters helped in my filter selection?" Q3) "I felt that I could select filters according to my own preference?" and an open question: Q4) "Which conditions did you like?" The participants filled out responded to Q1–3 based on a seven-point scale (strongly disagree = 1, neutral = 4, strongly agree = 7, and Q4: None condition = 1, neutral = 4, Proposed condition = 7).

8.3 Results

Table 3 shows the quantitative results of our user study on the two tasks. The None and Proposed conditions had similar results on average median selection times and filter comparisons. Since the reference ratios of the None and Proposed conditions were also similar, we confirmed that participants frequently selected filters with top-1 probability under both conditions. However, the reference ratio of task 1 was significant, which means the participants prioritized the top-1 filters under the Proposed condition. We also found significant selection ratios of the top-1 recommended filters. This means that participants selected the top-1 recommended filters more frequently.

Figure 8 shows histograms of selection distributions under the None and Proposed conditions. The histograms of each participant are similar under both conditions. We also computed histogram intersection as a metric of distribution similarity. The average result of all participants was 0.78 (SD: 0.08). Therefore, we confirmed that the distributions of participants' selections are similar under both conditions.

In task 2, the average selection ratio of the top-1 recommended filters was 0.735 (SD: 0.15). There is the significance for filters with the top-1 probability of prediction results and randomly selected filters. These results indicate that the proposed all-prediction algorithm is able to reflect participants' preferences over random recommendations.

Table 3. Results of our user study. * shows the statistical significance revealed by the Wilcoxon signed-rank test (significance level: $p < 0.05$)

Task 1	None condition	Proposed condition	p-value
selection times	9.34 (SD:0.36)	10.13 (SD:0.38)	0.1307
filter comparisons	5.19 (SD:1.84)	4.75 (SD:1.79)	0.0656
reference ratio of top-1	0.90 (SD:0.11)	0.93 (SD:0.10)	0.2721
first reference ratio of top-1	0.30 (SD:0.26)	0.40 (SD:0.27)	**0.0302***
selection ratio of top-1	0.33 (SD:0.14)	0.475 (SD:0.13)	**0.0028***
Task 2	Random filter	Top-1 filter (Proposed)	
Selection ratio	0.265 (SD:0.15)	**0.735 (SD:0.15)**	**0.0026***

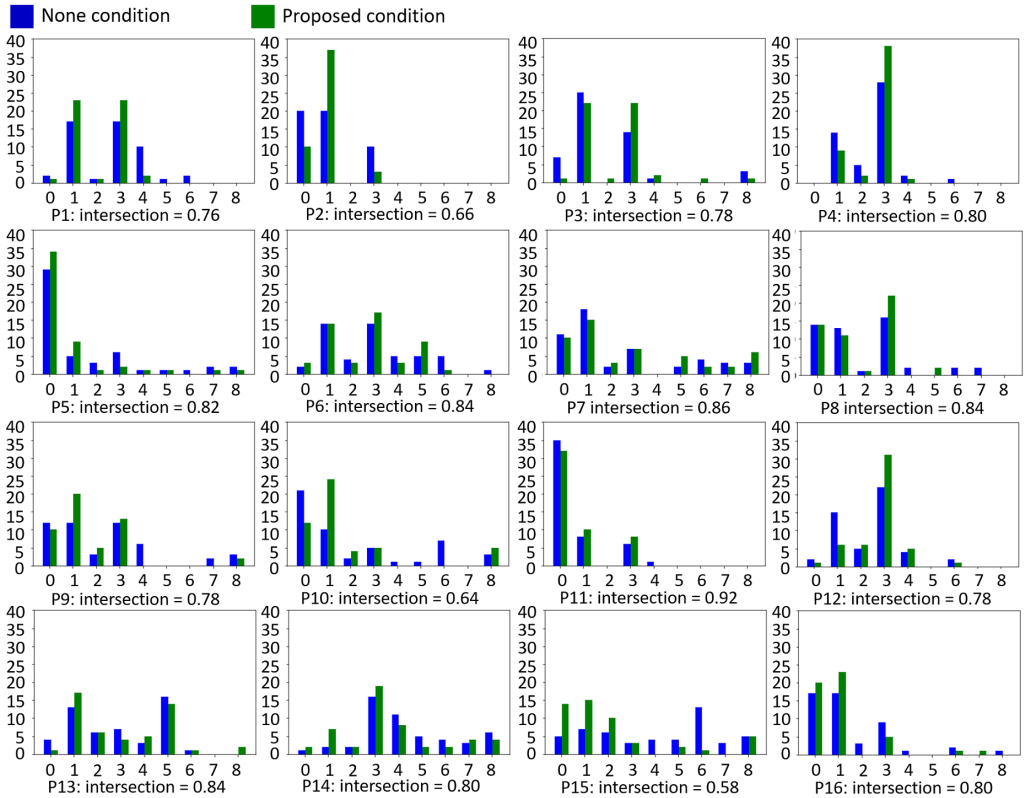

Fig. 8. Histogram comparison of None and Proposed conditions and histogram intersection values for each participant. Distributions of participants' selections were similar under both conditions. Average histogram intersections was 0.78 (SD:0.08)

Figure 9 shows the results of qualitative feedback. We confirmed that the participants gave positive feedback for the proposed all-prediction algorithm (Q1–3) and filter recommendations (Q4).

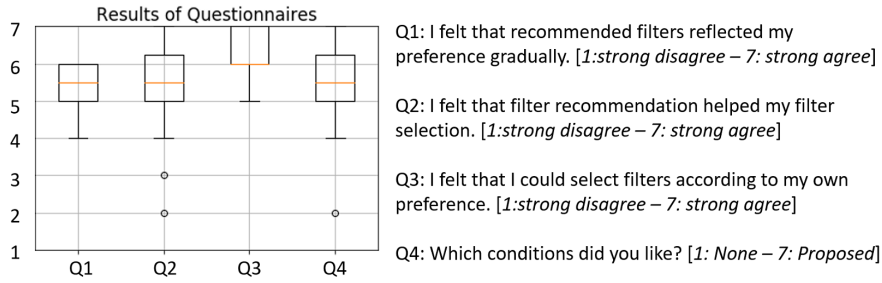

Fig. 9. Results of qualitative feedback. Participants answered questionnaire based on seven-point scale (Q1–3: strongly disagree = 1, neutral = 4, strongly agree = 7, and Q4: None condition = 1, neutral = 4, Proposed condition = 7). participants gave positive feedback on proposed all-prediction algorithm and filter recommendations.

9 DISCUSSION

9.1 Summary of Algorithm Evaluations

Through our series of algorithm evaluations, we confirmed that the proposed algorithms for both All and Limited Prediction settings performed better than baselines. In Evaluation 1, we compared our all-prediction algorithm with traditional supervised learning and bandit algorithms. The proposed algorithm outperformed the baselines in terms of top-1 accuracy and MMR as well as empirical regrets. It was able to quickly find and track an optimal policy for a target user. In contrast, the bandit algorithm could not detect the optimal policy in limited T time steps. The supervised learning algorithm with all-in-one training data exhibited comparable top-3 accuracy due to selection bias of image filters, *i.e.*, top-3 popular filters gathered many selections, as shown in Figure 4. However, this algorithm exhibited poor top-1 accuracy and MRR because the all-in-one data could not reflect individual preferences. In contrast, by changing the reward value R_{top3}, the performance of the proposed all-prediction algorithm could be tuned and outperformed the supervised learning algorithm regarding top-3 accuracy.

In evaluation 2, we confirmed that increasing the number of policies in a set improved prediction accuracy on an optimal policy and of the proposed all-prediction algorithm. As shown in Figures 4 and 8, user preferences of image filter selections were diverse, which resulted in different selection distributions among users. In contrast, a set may cover the preference of a target user by using a policy from another user with a similar preference if the set contains enough policies. The proposed all-prediction algorithm was thus able to find an optimal policy with a reasonable performance.

The results of Evaluation 3 for the limited-prediction algorithm indicate that this algorithm with the KL-UCB algorithm outperformed the competitive algorithms. Using the exponentially weighted average of prediction results, our algorithm was able to construct a personalized model using few policies from a set. The evaluation of processing time under various M showed that the prediction times of each policy affected to overall processing times rather than selecting M policies. When $M = 15$, the overall processing time was around 150 milliseconds (6.67 predictions per second). Although the processing time is acceptable for our image-filter-selection application, 15 or more prediction times may cause delay in response time on other interactive systems, which can make users uncomfortable. We thus conclude that reducing prediction times using the proposed limited-prediction algorithm is valuable for designing adaptive interactive systems.

9.2 Potential Effectiveness of Presenting Recommendation from Our Algorithm

The results of our pilot user study indicate the potential effectiveness of our recommendation interface with the proposed algorithms. We received positive feedback for the recommendations from our algorithms. From Q1 and Q2, many participants felt that the recommendation reflected their preference for image filters and helped in their selections. In Tasks 1 and 2, participants actively selected top-1 filters recommended from our algorithms. This indicates that the participants were able to rely on our recommendations for their image-filter selections.

We observed the potential of our algorithms to make user selection more efficient. In Task 1, the participants frequently started to review a filter candidate with top-1 recommendations. The participants also said they liked the Proposed condition (from Q4). We thus argue that the proposed algorithms give participants cues for complex decision making regarding image-filter selection. However, selection times and filter comparisons were similar under both conditions. It is necessary to improve representation methods and recommendation accuracy.

We also argue that the recommendations from our algorithms were able to preserve participants' preference for image-filter selections. All participants mentioned that they could select image filters based on their preferences (from Q3). We also observed that selection histograms from both None and Proposed conditions were similar for many participants, as shown in Figure 8. These facts also suggest that our algorithms successfully create personalized models for participants that reflect their personal preferences.

9.3 Limitations and Future Works

Prediction with large amount of policies. The main limitation of our study was the number of policies used in our algorithm evaluations and the user study. Our algorithm evaluations showed the advantages of the proposed algorithms using 300 image-filter selections from 16 participants. The number of participants was too small to definitively evaluate the proposed algorithms. We plan to evaluate our algorithms with a large number of policies in a set. However, it will be difficult to access all policies in a time step for practical interactive systems. We assume that the proposed algorithm for the Limited Prediction setting will be effective in such a scenario.

Theoretical Guarantees of Regret Bound. Previous studies on online learning (*e.g.*, bandit algorithms) have addressed the analysis of theoretical guarantees for regret bounds [4, 9, 15, 28, 34]. In contrast, our main focus was to empirically evaluate whether the proposed algorithms predict context-dependent user selections and assist their selection process. Although the evaluation showed the advantages of the proposed algorithms with our dataset, theoretical understanding is important to guarantee their performance.

Other Applications. Important future work is to apply the proposed algorithms to other applications that require users to make complex context-dependent decision making. We applied the proposed algorithms to an actual image-filter-selection application to evaluate their performance and investigated the effectiveness of the recommendations. Although we believe this study is valuable to show the effectiveness of the proposed algorithms, we will consider applying them to other types of applications such as food delivery services.

10 CONCLUSION

We proposed online learning algorithms that predict user selections based on personal preferences and given context information for assisting in complex context-dependent decision making. We introduced the combination of supervised learning from a dataset of other user selections and online learning with a set of pre-trained policies. We considered two types of the problem settings:

1) an online learner can access all prediction results from a set of policies (All Prediction setting), and 2) an online learner is allowed to access a limited number of predictions from a set (Limited Prediction setting). The algorithm for the All Prediction setting selects a policy with the current best weight for prediction then updates the weights of all policies based on user selection to quickly find an optimal policy in each time step. The algorithm for the Limited Prediction setting selects a limited number of policies from a set for combine prediction results then updates the weights and parameters of the selected policies in each time step. The limited-prediction algorithm computes the arm scores of policies based on bandit algorithms to address the exploration and exploitation problem for quickly building reasonable combinations of policies.

We applied the proposed algorithms to an image-filter-selection application. In this scenario, users were asked to select a suitable filter for a given image based on their preferences. This application can present the top-3 recommendations of image filters from the proposed algorithms. The algorithms sequentially learn target user preferences based on the features of the given images and user filter selections. We built a dataset on the image filter application to evaluate our algorithms. The evaluations showed that our algorithm for the All Prediction setting outperformed traditional supervised learning and bandit algorithms and improved prediction accuracy as the number of policies in a set increased. The evaluations also showed that our algorithm for the Limited Prediction setting had better prediction accuracy than state-of-the-art algorithms for a similar problem setting (*e.g.*, multi-armed bandit with additional observations). We also conducted a pilot user study to investigate the potential effectiveness of the recommendations based on the proposed algorithms. The results indicate that our recommendations reflect target user preferences and give participants cues to make satisfying filter selections.

ACKNOWLEDGMENTS

We thank all study participants. This work was sponsored in part by JST CREST (JPMJCR14E1), JST AIP-PRISM (JPMJCR18ZG), and JST SICORP (JPMJSC1605).

REFERENCES

[1] Charu C. Aggarwal. 2016. *Recommender Systems: The Textbook* (1st ed.). Springer Publishing Company, Incorporated.

[2] Eugene Agichtein, Eric Brill, Susan Dumais, and Robert Ragno. 2006. Learning User Interaction Models for Predicting Web Search Result Preferences. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '06)*. ACM, New York, NY, USA, 3–10. DOI: http://dx.doi.org/10.1145/1148170.1148175

[3] Christopher J Anderson. 2003. The psychology of doing nothing: forms of decision avoidance result from reason and emotion. *Psychological bulletin* 129, 1 (2003), 139.

[4] Peter Auer. 2003. Using Confidence Bounds for Exploitation-exploration Trade-offs. *J. Mach. Learn. Res.* 3 (March 2003), 397–422. http://dl.acm.org/citation.cfm?id=944919.944941

[5] Linas Baltrunas, Marius Kaminskas, Bernd Ludwig, Omar Moling, Francesco Ricci, Aykan Aydin, Karl-Heinz Lüke, and Roland Schwaiger. 2011. InCarMusic: Context-Aware Music Recommendations in a Car. In *E-Commerce and Web Technologies*, Christian Huemer and Thomas Setzer (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 89–100.

[6] Björn Brodén, Mikael Hammar, Bengt J. Nilsson, and Dimitris Paraschakis. 2018. Ensemble Recommendations via Thompson Sampling: An Experimental Study Within e-Commerce. In *23rd International Conference on Intelligent User Interfaces (IUI '18)*. ACM, New York, NY, USA, 19–29. DOI: http://dx.doi.org/10.1145/3172944.3172967

[7] Nicolò Cesa-Bianchi, Yoav Freund, David Haussler, David P. Helmbold, Robert E. Schapire, and Manfred K. Warmuth. 1997. How to Use Expert Advice. *J. ACM* 44, 3 (May 1997), 427–485. DOI: http://dx.doi.org/10.1145/258128.258179

[8] Nicolo Cesa-Bianchi and Gabor Lugosi. 2006. *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA.

[9] Olivier Chapelle and Lihong Li. 2011. An Empirical Evaluation of Thompson Sampling. In *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2249–2257. http://papers.nips.cc/paper/4321-an-empirical-evaluation-of-thompson-sampling.pdf

[10] Xiangmin Fan, Youming Liu, Nan Cao, Jason Hong, and Jingtao Wang. 2015. MindMiner: A Mixed-Initiative Interface for Interactive Distance Metric Learning. In *Human-Computer Interaction – INTERACT 2015*, Julio Abascal, Simone Barbosa, Mirko Fetter, Tom Gross, Philippe Palanque, and Marco Winckler (Eds.). Springer International Publishing, Cham, 611–628.

[11] Leah Findlater and Jacob Wobbrock. 2012. Personalized Input: Improving Ten-finger Touchscreen Typing Through Automatic Adaptation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 815–824. DOI: http://dx.doi.org/10.1145/2207676.2208520

[12] James Fogarty, Desney Tan, Ashish Kapoor, and Simon Winder. 2008. CueFlik: Interactive Concept Learning in Image Search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. 29–38. DOI: http://dx.doi.org/10.1145/1357054.1357061

[13] Johannes Frnkranz and Eyke Hllermeier. 2010. *Preference Learning* (1st ed.). Springer-Verlag, Berlin, Heidelberg.

[14] Krzysztof Z. Gajos, Daniel S. Weld, and Jacob O. Wobbrock. 2010. Automatically generating personalized user interfaces with Supple. *Artificial Intelligence* 174, 12 (2010), 910 – 950. DOI: http://dx.doi.org/https://doi.org/10.1016/j.artint.2010.05.005

[15] Aurélien Garivier and Olivier Cappé. 2011. The KL-UCB algorithm for bounded stochastic bandits and beyond. In *Proceedings of the 24th annual Conference On Learning Theory*. 359–376.

[16] Satyen Kale. 2014. Multiarmed bandits with limited expert advice. In *Conference on Learning Theory*. 107–122.

[17] Yuki Koyama, Daisuke Sakamoto, and Takeo Igarashi. 2014. Crowd-powered Parameter Analysis for Visual Design Exploration. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 65–74. DOI: http://dx.doi.org/10.1145/2642918.2647386

[18] Yuki Koyama, Daisuke Sakamoto, and Takeo Igarashi. 2016. SelPh: Progressive Learning and Support of Manual Photo Color Enhancement. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 2520–2532. DOI: http://dx.doi.org/10.1145/2858036.2858111

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 1097–1105. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[20] Seok Kee Lee, Yoon Ho Cho, and Soung Hie Kim. 2010. Collaborative filtering with ordinal scale-based implicit ratings for mobile music recommendations. *Information Sciences* 180, 11 (2010), 2142 – 2155. DOI: http://dx.doi.org/https://doi.org/10.1016/j.ins.2010.02.004

[21] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2010. A Contextual-bandit Approach to Personalized News Article Recommendation. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. ACM, New York, NY, USA, 661–670. DOI: http://dx.doi.org/10.1145/1772690.1772758

[22] G. Linden, B. Smith, and J. York. 2003. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (Jan 2003), 76–80. DOI: http://dx.doi.org/10.1109/MIC.2003.1167344

[23] Jiming Liu, Chi Kuen Wong, and Ka Keung Hui. 2003. An adaptive user interface based on personalized learning. *IEEE Intelligent Systems* 18, 2 (Mar 2003), 52–57. DOI: http://dx.doi.org/10.1109/MIS.2003.1193657

[24] Abhinav Mehrotra, Robert Hendley, and Mirco Musolesi. 2016. PrefMiner: Mining User's Preferences for Intelligent Mobile Notification Management. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '16)*. ACM, New York, NY, USA, 1223–1234. DOI: http://dx.doi.org/10.1145/2971648.2971747

[25] Pablo Messina, Vicente Dominguez, Denis Parra, Christoph Trattner, and Alvaro Soto. 2019. Content-based artwork recommendation: integrating painting metadata with neural and manually-engineered visual features. *User Modeling and User-Adapted Interaction* 29, 2 (01 Apr 2019), 251–290. DOI: http://dx.doi.org/10.1007/s11257-018-9206-9

[26] Lijing Qin, Shouyuan Chen, and Xiaoyan Zhu. *Contextual Combinatorial Bandit and its Application on Diversified Online Recommendation.* 461–469. DOI: http://dx.doi.org/10.1137/1.9781611973440.53

[27] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*. 285–295. DOI: http://dx.doi.org/10.1145/371920.372071

[28] Yevgeny Seldin, Peter Bartlett, Koby Crammer, and Yasin Abbasi-Yadkori. 2014. Prediction with Limited Advice and Multiarmed Bandits with Paid Observations. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32 (ICML'14)*. JMLR.org, I–280–I–287. http://dl.acm.org/citation.cfm?id=3044805.3044838

[29] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* abs/1409.1556 (2014). http://arxiv.org/abs/1409.1556

[30] WILLIAM R THOMPSON. 1933. ON THE LIKELIHOOD THAT ONE UNKNOWN PROBABILITY EXCEEDS ANOTHER IN VIEW OF THE EVIDENCE OF TWO SAMPLES. *Biometrika* 25, 3-4 (12 1933), 285–294. DOI: http://dx.doi.org/10.

1093/biomet/25.3-4.285

[31] John Tierney. 2011. Do you suffer from decision fatigue. *The New York Times* (2011).

[32] Aaron van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2643–2651. http://papers.nips.cc/paper/5004-deep-content-based-music-recommendation.pdf

[33] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How Transferable Are Features in Deep Neural Networks?. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'14)*. MIT Press, Cambridge, MA, USA, 3320–3328. http://dl.acm.org/citation.cfm?id=2969033.2969197

[34] Donggyu Yun, Alexandre Proutiere, Sumyeong Ahn, Jinwoo Shin, and Yung Yi. 2018. Multi-armed Bandit with Additional Observations. *Proc. ACM Meas. Anal. Comput. Syst.* 2, 1, Article 13 (April 2018), 22 pages. DOI: http://dx.doi.org/10.1145/3179416

[35] Hengshu Zhu, Enhong Chen, Hui Xiong, Kuifei Yu, Huanhuan Cao, and Jilei Tian. 2014. Mining Mobile User Preferences for Personalized Context-Aware Recommendation. *ACM Trans. Intell. Syst. Technol.* 5, 4, Article 58 (Dec. 2014), 27 pages. DOI: http://dx.doi.org/10.1145/2532515