Automated Mechanism to Choose Between Reinforcement Learning and Contextual Bandit in Website Personalization

Abhimanyu Mitra
AMitra@walmartlabs.com
Walmart Labs
Sunnyvale, California

Afroza Ali
AAli@walmartlabs.com
Walmart Labs
Sunnyvale, California

Xiaotong Suo
Xiaotong.Suo@walmartlabs.com
Walmart Labs
Sunnyvale, California

Kailing Wang
Kailing.Wang@walmartlabs.com
Walmart Labs
Sunnyvale, California

Kannan Achan
KAchan@walmartlabs.com
Walmart Labs
Sunnyvale, California

ABSTRACT

Personalizing customer experience typically requires a range of experiments with an ever-changing content pool and evolving customer preferences. Since pure experimentation is costly, we need an appropriate explore-exploit strategy to complement the personalization models and allow the models to perform the necessary experimentation. If our goal in choosing the content for each customer is to get the best long-term value from them, our explore-exploit strategy should also aim to exploit for the same. With this exploitation goal, the natural choice of an explore-exploit strategy seems to be a reinforcement learning framework, which optimizes for long-term rewards for every state (appropriate customer segment) of the relevant Markov chain. However, one of the crucial challenges of a reinforcement learning framework is the limited number of available trials from which it has to estimate the long-term reward for every state-content pair, since the content pool is constantly changing. Since a good simulation model for customer preferences is not available, it is costly to experiment with real customers by increasing the number of trials for the framework to learn, potentially providing sub-optimal customer experiences. On the other hand, if short-term rewards are indicative of long-term rewards, once learning is complete, a reinforcement learning framework and a contextual multi-armed bandit [2, 7] framework (which uses the states of the Markov chain as context) might make very similar choices of contents for each of the states. Moreover, since a contextual bandit framework tries to learn only short-term reward as opposed to a long-term reward, the learning happens faster. Thus, in some of the modules of the website, it might be more cost-effective to run a contextual bandit algorithm as an explore-exploit strategy, even when we want to optimize long-term reward from our customers. We propose a framework which analyzes historical data on customer-content interaction to predict for each module and content pool, which strategy would work best there: reinforcement learning or contextual bandit. We show with empirical data that our framework predicts the correct explore-exploit strategy with a high precision rate. Furthermore, our framework quickly adapts its prediction even when the optimal explore-exploit strategy changes with time within the same module.

IRS 2020, August 24, 2020, Virtual,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

CCS CONCEPTS

• **Mathematics of computing → Probabilistic inference problems**; • **Information systems → Personalization**; • **Applied computing → electronic commerce**.

KEYWORDS

Reinforcement Learning, Contextual Multi-Armed-Bandit, Website Personalization, Explore-Exploit.

ACM Reference Format:
Abhimanyu Mitra, Afroza Ali, Xiaotong Suo, Kailing Wang, and Kannan Achan. 2020. Automated Mechanism to Choose Between Reinforcement Learning and Contextual Bandit in Website Personalization. In *Proceedings of 1st International Workshop on Industrial Recommendation Systems at SIGKDD'20 (IRS 2020).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Website personalization requires redesigning the webpage according to the customers and their implicit or explicit preferences, which translates to picking the right content from a pool of contents. These contents could be the web modules which paint the page for the customer, or the contents/items/pictures that fill a web module. In other words, we can think of personalization as choosing the right content(s) for the free web slot(s) from a pool of contents, and if multiple slots are available for the same pool, for example, multiple slots within the same module which could be filled with contents from the same pool, deciding the order of the chosen contents according to customers and their implicit or explicit preferences. Typically, personalization algorithms maximize a reward, for example, clicks or revenue, which means the choice of contents or ordering of the contents is driven by maximizing rewards for customers. As opposed to chasing the maximum reward from customers in a single web session, there is an increasing interest in attaining higher long-term customer value through personalizing the contents of a webpage. In other words, there is an increasing interest in personalization algorithms that optimize long-term rewards from the

customers, which can be computed by suitably aggregating current session rewards with that of several future web sessions of the same customer.

For a website, with time, the content pools for each of the web slots keep changing and so do customer preferences. Therefore, predicting the right content for each customer requires the personalization algorithms to experiment with new contents to estimate the reward associated with each of the customer-content pairs. With a live website and the ever-changing content pools, it is impossible to run pure exploration as it is too costly. So we need an explore-exploit strategy to complement our personalization algorithms in order to perform the necessary explorations. In such a case, the exploit part of an explore-exploit strategy should be designed to optimize a reward, which is similar to the reward that the corresponding personalization algorithm optimizes. Thus, if a personalization algorithm is trained to optimize for long-term reward, the corresponding explore-exploit strategy should do the same. A natural explore-exploit strategy to exploit long-term rewards seems to be a reinforcement learning framework [4] which traces influences of content shown to a customer by noting transformation of the customer in future sessions and optimizes the choice of content based on long-term rewards, where the long-term reward for each customer-content interaction is computed by aggregating current session rewards with discounted future session rewards from the same customer.

To simplify the reinforcement learning framework, we assume mutually exclusive customer segments are available to us, which we could use as the states of the associated Markov chain. To make the state space complete, we add two states, one is where a visiting customer does not belong to any of the segments and another where the customer does not return to the website following a visit. Though many research has shown how to build an appropriate state space suitable for reinforcement learning framework, including deep reinforcement learning networks, see for example [8], time constraints force us to restrict ourselves to such a simplified version. However, we believe the following discussion will be as relevant even when the state space is more diligently and appropriately constructed.

In the literature on Reinforcement Learning, the primary application is training computers to play games [3, 9–11], where they are trained to anticipate several steps ahead in order to finally win the game. In our application, we want to choose contents to get the highest long-term rewards from our customers and this application brings a few additional complications. In a game, the next step is observed at a certain interval unless the game ends, whereas in our application the next step only comes when the customer returns to the webpage and therefore happens after a random time interval (if the customer never returns, which is also a possible state and only after waiting for a long-enough interval, we would be able to conclude if a customer's state has changed to this one). So, in building our reinforcement learning framework, this additional randomness needs to be accounted for. Moreover, when computers are trained to play games, the learning phase involves computers playing with each other and therefore, more trials for computers to learn just means additional simulations. On the other hand, we do not have a good simulation model to simulate customer-content interaction and our reinforcement learning framework can only learn by trying contents with customers on a live website and therefore, while the framework is learning, we are potentially providing the customers with a sub-optimal website experience, leading to a lot more expensive trials.

Any explore-exploit strategy starts from pure exploration and as it learns the rewards associated with each action for each context, it starts to exploit. Thus, such a strategy can only bring benefits over a pure exploration strategy, once it has learnt enough. To get most from the reinforcement learning framework, it first needs to learn for each context, which content produces the highest long-term reward. Since the learning happens with each trial, which is when a customer visits the webpage, we cannot expedite the process of learning through simulation or other methods. Moreover, since the content pool is only active on the webpage for a short time, if the framework takes too long to learn, the same content pool might no longer be active when the framework is ready to exploit. Therefore, the need arises to figure out alternatives, which might learn faster and make similar content choices as the reinforcement learning model would, once it had learnt the best action for each state. One such alternative is to rely on short-term rewards if the relative comparison of short-term rewards for different contents in the content pool is very similar to that of long-term rewards. For example, say, for each state, the content bringing the best long-term reward also brings in the best short-term reward. Then both methods would choose the same content once they are ready to exploit. Relying on short-term rewards brings us to a contextual multi-armed bandit model, where we use the states of the Markov chain as contexts, which acts as the cheaper alternative compared to enabling a reinforcement learning model, since it learns faster but is equally effective in this case. In fact, this could be a better option since it wastes lesser time (and thus lesser bad customer experiences throughout the exploration period) in learning which actions work best in which context.

The choice of the two alternatives described above: the reinforcement learning model or the contextual multi-armed bandit model as an explore-exploit strategy, depends on the content pool and the set of states. Even if we are making the choice between the two models for the same web module, as the content pool changes over time, the optimal choice could also change. Customers' states might also change over time because of dynamic nature of their preferences. Therefore, even for the same web module, the optimal choice could be changing over time. Of course, the choice could be be different for two different modules on the same webpage. Therefore, we need an automated mechanism which crunches historical data on customer-content interaction to predict at a regular interval (maybe every day) for each of the web modules, given their current content pools, which explore-exploit strategy is more appropriate for the module. Similarly, if we are using an explore-exploit strategy to choose and rank web modules on a webpage, the automated mechanism should predict for each of the webpages, given its current set of possible web modules, which model is more appropriate: reinforcement learning or contextual multi-armed bandit.

In this paper, we describe how we build such an automated mechanism. In Section 2, we show how the automated mechanism works. In Section 3, we present empirical results. In Section 4, we conclude with future research.

2 METHOD

2.1 Motivation

To motivate the use of enabling a Reinforcement Learning (RL) framework to complement a Contextual Multi-Armed-Bandit(CMAB), we first describe how these two frameworks are used in website personalization. For every customer visit, personlization models generate a list of eligible content candidates per module and CMAB or RL framework ranks the final contents before it is rendered on the web module. This system renders content for every module on the homepage and item pages. A CMAB framework makes the exploration-exploitation balance separately for each context. Customers have been grouped based on their interactions with the website and other relevant activities, and we use them as contexts for the CMAB framework. For example, a customer who have shown strong affinity towards technology products, are grouped under 'Tech' customer segment, and an another customer, identified to be part of busy families, is classified as 'Busy Families' etc. The short-term reward function, which the CMAB framework tries to estimate and optimize, could be engagement metrics such as clicks on the web module, or conversion metrics if the customer purchases products featured in the web module. However, the fact that CMAB framework never takes a long-term view or connects customer activities over multiple web sessions can lead to inaccurate results. For example, suppose it is established through statistically significant metrics that 'Busy families' customers are less likely to engage in a 'Tech' content and therefore, in this case, a CMAB framework never shows 'Tech' content to customers identified as 'Busy families'. This choice might be myopic if there is a probability that a 'Busy families' customer if presented with 'Tech' content, might get interested in 'Tech' products and thus transform to a 'Tech' customer in future. An RL framework mitigates this problem by connecting multiple web sessions of the same customer and also incorporating the influence of contents in transitioning a customer to belong to a different group/context in some future web session into the model. Thus, an RL model forego some short-term rewards (say, in-session engagement) in the hope of getting higher rewards in future.

2.2 Detailed description of our method

We now formally introduce our notation. We denote the customer groups as the states of a Markov chain $\{S_1, S_2, \cdots, S_n\}$, and assume that a customer can only belong to one state at a given time. For each state, action space consists of eligible contents; i.e., for each state S_j, $j = 1, 2, \cdots, n$, there are possible actions $\{C_{j,1}, C_{j,2}, \cdots, C_{j,K_{S_j}}\}$, where K_{S_j} is the total number of eligible candidate contents for the state S_j. Short-term reward function can be clicks on the web module, or conversion metrics. In details, if the short-term reward function is defined as clicks on the web module, and if the current state of a customer is S_i, $i \in \{1, 2, \cdots, n\}$, when (s)he visits webpage and sees impression, the short-term reward r_i is 1 if (s)he clicks on the module impression and 0 otherwise. To calculate the long-term reward associated with the impression, we need to track the customer over future sessions and the rewards on the module impressions from those sessions. Suppose the customer comes back after $k_1 < k_2 < \cdots$ days from when (s)he first saw the impression and in those future web sessions, (s)he sees the module impressions $i_{k_1}, i_{k_2}, \cdots$ in the chronological order and the corresponding rewards to the module impressions $i_{k_1}, i_{k_2}, \cdots$ are $r_{i_{k_1}}, r_{i_{k_2}}, \cdots$ respectively, then the long-term reward of the impression can be computed as

$$q_i = r_i + \gamma^{k_1} r_{i_{k_1}} + \gamma^{k_2} r_{i_{k_2}} + \cdots, \tag{1}$$

where $0 < \gamma < 1$ is a fixed discount factor used to discount future rewards. To approximate the infinite series, we ignore terms when the discount factor becomes small enough, say less than ϵ, where ϵ is a positive number. Let T be the smallest integer so that $\gamma^T < \epsilon$. Then for all $k_n > T$, $\gamma^{k_n} < \epsilon$. Let m be the largest integer so that $k_m \leq T$. Then, q_i in (1) could be approximated as

$$q_i = r_i + \gamma^{k_1} r_{i_{k_1}} + \gamma^{k_2} r_{i_{k_2}} + \cdots + \gamma^{k_m} r_{i_{k_m}}. \tag{2}$$

The short-term reward r_i and long-term reward q_i for each impression could be averaged over the state, action pairs to obtain average short-term reward $R(S_j, C_{j,k})$ and average long-term reward $Q(S_j, C_{j,k})$ for each state, action pair $(S_j, C_{j,k})$. Using these average estimates, we could approximate q_i in (1) using one-step updates

$$q_i = r_i + \gamma^{k_1} Q(S_{i_{k_1}}, C_{i_{k_1}, i_{k_1}}), \tag{3}$$

where $S_{i_{k_1}}$ is the state of the customer when (s)he returns first time after k_1 days and views impression i_{k_1} and $C_{i_{k_1}, i_{k_1}}$ is the chosen content for the module impression i_{k_1}.

In the CMAB framework, all we need is to update the short-term reward $R(\cdot, \cdot)$ by simply including new impressions in the average. In the RL framework, we use (3) to estimate long-term reward associated with each impression. It is an online learning framework where we begin with an initialization $Q(\cdot, \cdot) \equiv R(\cdot, \cdot)$ and as more impressions come in (potentially customers come back), we update the long-term rewards associated with each impression using (3) and then average over state-action pairs to update estimates of $Q(\cdot, \cdot)$. As we collect more data and $R(\cdot, \cdot)$ and $Q(\cdot, \cdot)$ are averages over a lot of impressions, by Central Limit theorem, we could assume they would follow Normal distribution and the variance of the Normal distribution could be estimated via the short-term rewards r_i-s associated with each impression and long-term rewards q_i-s associated with each impression (approximated via (3)).

For each state S_j, $j = 1, 2, \cdots n$, the CMAB framework allocates impressions to $\{C_{j,1}, C_{j,2}, \cdots, C_{j,K_{S_j}}\}$ in proportion to estimated $R(S_j, C_{j,k})$ and the RL framework allocates impressions to $\{C_{j,1}, C_{j,2}, \cdots, C_{j,K_{S_j}}\}$ in proportion to estimated $Q(S_j, C_{j,k})$. However, adapting this to other mechanisms of resolving explore-exploit dilemma like Thompson sampling [1, 6] follows similar steps. For example, we can run simulations to figure out impression allocation using any mechanism resolving explore-exploit dilemma, as long as the mechanisms are solely reliant on distributions of $R(S_j, C_{j,k})$ and $Q(S_j, C_{j,k})$, $j = 1, 2, \cdots, n, k = 1, 2, \cdots, K_{S_j}$.

For each state S_j, $j = 1, \cdots n$, note that the only way these two frameworks resulting in different customer experiences is through allocating different impressions to contents $\{C_{j,1}, C_{j,2}, \cdots, C_{j,K_{S_j}}\}$. Therefore, to compare the impact of a RL framework over CMAB framework, we need to figure out how these different impression allocations are bringing in different total long-term rewards. In

absence of an A/B test, all we have is historical data of impressions and their short-term rewards. We divide the historical impression data in training and evaluation period so that chronologically the evaluation period comes after the training period. From the training data we estimate $R(\cdot,\cdot)$ and $Q(\cdot,\cdot)$ and assume that in the evaluation period, for each S_j, our CMAB framework allocates impressions to $\{C_{j,1}, C_{j,2}, \cdots, C_{j,K_{S_j}}\}$ in proportion to the estimated $R(S_j, C_{j,k})$ and our RL framework allocates impressions to $\{C_{j,1}, C_{j,2}, \cdots, C_{j,K_{S_j}}\}$ in proportion to estimated $Q(S_j, C_{j,k})$.

To connect the impression allocations with the total reward, we need to know the average long-term reward per impression associated with each state-action pair in the evaluation period. To estimate that, we use the long-term reward estimate of each impression in the evaluation period using (2) and then average them over the state-action pairs to get $\bar{q}(S_j, C_{j,k})$ for each state-action pair $(S_j, C_{j,k})$. Since $\bar{q}(S_j, C_{j,k})$ is also an average, the distribution of $\bar{q}(S_j, C_{j,k})$ could be approximated by Normal distribution and the variance can be estimated using the q_i-s associated with each impression, as given in (2).

Let $W^R(S_j, C_{j,k})$ denote the number of allocated impressions to $C_{j,k}$ in state S_j using our CMAB framework and $W^Q(S_j, C_{j,k})$ denote the same using our RL framework. Note that the condition $\sum_k W^R(S_j, C_{j,k}) = \sum_k W^Q(S_j, C_{j,k})$ must be satisfied for all S_j, since total impressions to state S_j in the evaluation period is fixed. Let us also denote the total rewards associated with a CMAB framework T^R and a RL framework T^Q in the evaluation period could be computed as

$$T^R = \sum_j \sum_k W^R(S_j, C_{j,k}) \bar{q}(S_j, C_{j,k}) \tag{4}$$

$$T^Q = \sum_j \sum_k W^Q(S_j, C_{j,k}) \bar{q}(S_j, C_{j,k}). \tag{5}$$

Therefore the improvement in total reward from our RL framework over our CMAB framework could be computed as $(T^Q - T^R)/T^R$. We can also estimate the noise of this measured lift from simulation. As noted before, the estimates of $R(S_j, C_{j,k})$, $Q(S_j, C_{j,k})$ and $\bar{q}(S_j, C_{j,k})$ all follow Normal distribution with known mean and standard deviation. In each run of the simulations, we can simulate $R(S_j, C_{j,k})$, $Q(S_j, C_{j,k})$ and $\bar{q}(S_j, C_{j,k})$ from their respective distributions. Using these simulated numbers, we can obtain $W^R(S_j, C_{j,k})$ and $W^Q(S_j, C_{j,k})$, and then calculate T^R and T^Q and thereafter get a lift. Each simulation run gives us one estimated lift. Therefore, from multiple simulations, we can obtain the estimated lift through averaging as well as the noise associated with it.

2.3 Design Overview

The explore exploit system is tightly weaved into our personalization platform. The end-to-end system is illustrated in the figure 1. It comprises of 2 main components:

2.3.1 Optimal strategy selection and Model parameters estimation. We have built a complex evaluation system that comprises of a suit of frameworks to evaluate different reward strategies(global, contextual or RL) and pick the optimal strategy for each module independently . These module specific strategy recommendations are then used by specific parameter estimation models, to estimate the parameters of the reward functions for different modules.

2.3.2 Content candidates selection & Ranking. This is a real-time system that uses customer and content features to dynamically select a pool of eligible content candidates. Using the explore exploit reward parameters estimation for each of them , it ranks them in the descending order of their posterior probability weights. This final ranked list of content candidates for each recommendations module goes to the front end for rendering on customers' devices.

3 EXPERIMENTAL RESULTS

3.1 Setup

The offline evaluation framework predicts optimal reward strategy and is capable to evaluate for rewards based on user engagement, conversion or revenue – in this implementation, we have used click as the reward. For each module, the mean rewards are estimated for each pair of customer context as state, say S_j and content candidate as action, say $C_{j,k}$ denoted by $(S_j, C_{j,k})$.

We maintain a daily update model for customer segmentation. It uses customers' activity data and content level features to assign each customer to a segment or context. The model learns their evolving behavior, and update their segments/context daily.

The framework independently makes daily predictions of optimal strategy for different recommendation modules. Each prediction involves 2 chronological steps – training period and evaluation period.

3.1.1 Training Period. In this implementation, the evaluation framework uses historical click and impression data generated across several customers visits on the website. Our back end system records every click and visit data for each recommendation module and aggregates them for each day separately. During each training period which comprises of several training days, CMAB and RL models consume this data to update the model reward estimations for all state-content pairs $(S_j, C_{j,k})$.

CMAB based model estimates the parameters of the distribution of the mean short-term rewards function $R(\cdot,\cdot)$ for every state-content pair $(S_j, C_{j,k})$, each of which is approximated by a Normal distribution. For estimating long-term rewards, RL based model updates the parameters of the distribution of $Q(\cdot,\cdot)$, which is again approximated by Normal distribution for every state-content pair $(S_j, C_{j,k})$.

RL based model computes discounted long-term rewards for each repeat visit by a returning customer and updates the rewards of the last impression by the same customer using (3). For each state-content pair $(S_j, C_{j,k})$, these updated rewards are used to update the parameters of the distribution of $Q(S_j, C_{j,k})$. Initially, $Q(\cdot,\cdot)$ will be very similar to short-term reward distributions $R(\cdot,\cdot)$, however overtime, as the customers come back to the website, it updates $Q(\cdot,\cdot)$ with average discounted long-term rewards. We use the posterior sampling method called Thompson sampling to resolve the explore-exploit dilemma for both CMAB and RL based models.

Ideally the discounted long-term rewards can be awarded to infinitely long window of time in the past. Customers impressions

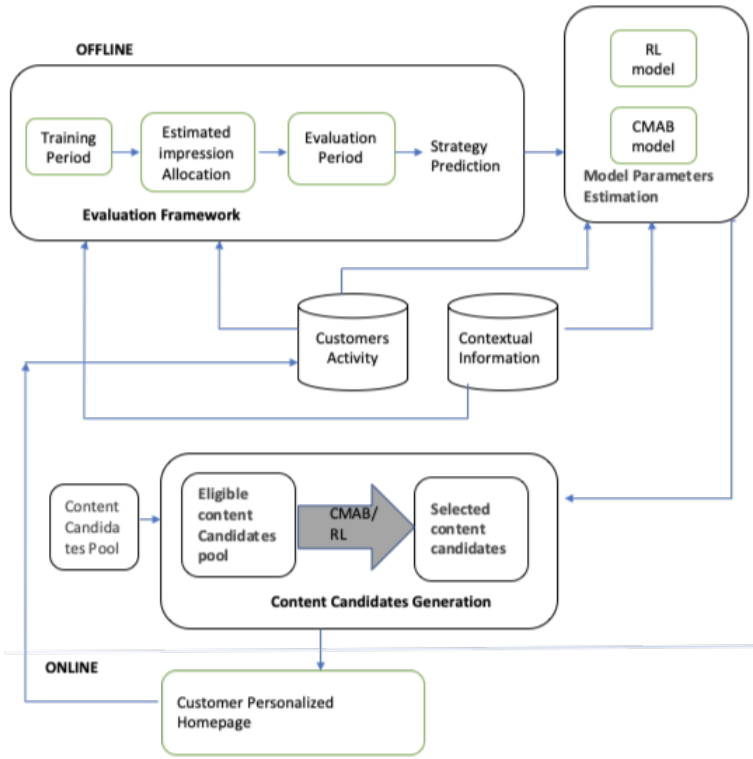

Figure 1: End-to-end design scheme of personalized online recommendation system. Its comprises of 3 main components, a)Evaluation framework for predicting the most reward generating model(CMAB/RL) per recommendation module, b)Model parameters estimation used for parameters estimation for each module based on the recommendation by the evaluation framework, c)Content Candidates Generation that extracts eligible content candidates using personalized models and select and order the final recommendations using RL/CMAB model parameters.

from months or years ago can be used to update the reward estimations of state-content pair $(S_j, C_{j,k})$. However, e-commerce world customers' preferences evolve and content space is changing quickly, long-term rewards beyond a certain window of time becomes insignificant or unimportant. To address this, the evaluation framework uses (2) and defines parameter T, which represents the window of time up to which the model computes discounted long-term rewards. The choice of the value of T may vary across different products and recommendation modules. It may also be impacted by how fast the content candidate pool changes, as well as on the definition of the reward function.

We use an exploration-biased modified Thompson Sampling as our impression allocation strategy for both RL and CMAB, where after sampling from posterior distributions of rewards, instead of sorting and picking the action/content with the highest sample, we use weighted random sampling using the generated samples as weights to pick the content. As an approximation of this impression allocation strategy, we can assume CMAB and RL models following this impression allocation strategy would allocate impressions to content space $\{C_{j,1}, C_{j,2}, \cdots, C_{j,K_{S_j}}\}$ in proportion to their reward functions $R(S_j, C_{j,k})$ and $Q(S_j, C_{j,k})$ respectively. For each state-content pair $(S_j, C_{j,k})$, the evaluation framework computes impression allocation weight, $W^R(S_j, C_{j,k})$ for CMAB and $W^Q(S_j, C_{j,k})$ for RL model.

3.1.2 Evaluation Period. For a given recommendation module, the aim of the evaluation framework is to select the model that maximizes expected cumulative long-term reward. The long-term reward is optimal when the model estimates higher impression allocation for state-content pair $(S_j, C_{j,k})$ with larger average long-term

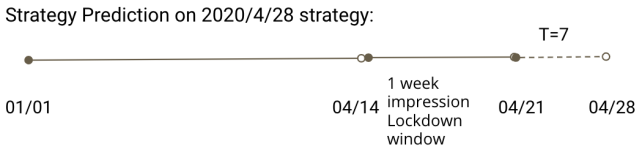

Figure 2: An example of training and evaluation period with T set to 7 and prediction date is April 28th 2020. The evaluation framework uses Jan 1st to April 13th as the training period to learn the impression allocations $W^R(S_j, C_{j,k})$ and $W^Q(S_j, C_{j,k})$ for each state-action pair $(S_j, C_{j,k})$ for CMAB and RL respectively. Following that, it runs the evaluation period from April 14th to April 27th to compute the average long-term rewards for every state-content pair $(S_j, C_{j,k})$. It updates the long-term reward using (2) for every impression in the lockdown period between April 14th to April 20th.

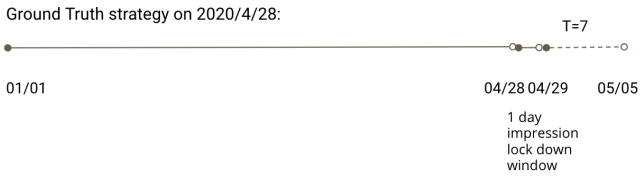

Figure 3: To generate the ground label strategy on April 28th, the evaluation framework used all the click-impression data from Jan 1st to April 27th and estimates the impression allocations for each state-action pair $(S_j, C_{j,k})$ for CMAB and RL models. It then estimates the average long-term reward for state-action pair $(S_j, C_{j,k})$ using the all the impressions in the impression lock down window, which is the prediction day and T days sliding window for each impression.

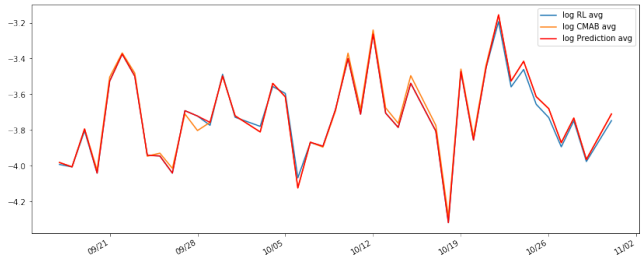

Figure 4: Time series plot of the daily prediction made by the evaluation framework for module A during Sept/Oct time window. In the graph, for each prediction date we have plotted logarithm of average cumulative long-term rewards T^R and T^Q for CMAB and RL. The blue curve represents expected total long-term rewards by RL model. The orange curve represents expected total long-term rewards by CMAB model. Optimal strategy prediction by the evaluation framework is shown by the red curve. The evaluation framework takes the expected cumulative long-term reward of the predicted optimal strategy. During this time window, the prediction curve often overlaps with T^Q shown by blue curve. For this analysis the value of the parameter T is set as 7.

reward. The evaluation period consists of impression lock down window and T days rolling window. Using every impression in the lock down window, the framework computes the average long-term reward for every state-action pair $(S_j, C_{j,k})$ using (2). An impression lock down window could be few days, in this implementation we have used 1 week following the training period. All the impressions in the impression lock down window uses T days of rolling window, where the value of T is the same as used in training the RL model. Each customer impression in the lock down period receives discounted long-term reward when the same customer shows up again during the rolling window period. For each state-action pair $(S_j, C_{j,k})$, these long-term rewards are then used to estimate parameters of the distribution of $\bar{q}(S_j, C_{j,k})$, which is approximated by a Normal distribution.

In this implementation, the total time window for the evaluation period is one week + T days.

Note that, using the impression allocation weights $W^R(S_j, C_{j,k})$ and $W^Q(S_j, C_{j,k})$ from the training period, and average long-term reward $\bar{q}(S_j, C_{j,k})$ for each state-content pair $(S_j, C_{j,k})$, the evaluation framework computes the expected cumulative long-term reward estimates by CMAB and RL models using (4) and (5) respectively. To estimate noise associated with them, the evaluation framework runs thousands of simulations – in each simulation, it samples from $R(S_j, C_{j,k})$, $Q(S_j, C_{j,k})$ and $\bar{q}(S_j, C_{j,k})$ from their respective Normal distributions to compute T^R and T^Q. Using the T^R and T^Q from each simulation, we could estimate the average lift $(T^Q - T^R)/T^R$ as well as the noise associated with the lift.

The evaluation framework chooses the model that generates largest expected cumulative long-term reward. Each point in the time series in Fig 4 and 5 are the average cumulative long-term rewards T^R and T^Q.

3.2 Performance

3.2.1 Prediction. The day following the evaluation period is the prediction day. The evaluation framework predicts the optimal reward strategy using impression allocations estimated in the training period and estimated average long-term reward for each state-content pair $(S_j, C_{j,k})$ in the evaluation phase.

3.2.2 Prediction Accuracy. To measure the prediction accuracy of the evaluation framework, we need the ground truth labels for optimal strategy. The optimal strategy, on a given day, is the one that generates maximum expected cumulative long-term reward. We use (4) and (5) to compute expected cumulative long-term reward estimation for CMAB and RL models, and then pick the maximum.

For ground truth labels, we first need to estimate how RL and CMAB models would allocate impressions on the prediction day. For each state-action pair $(S_j, C_{j,k})$, we use the historical data up to the prediction date in order to estimate the impression allocations $W^R(S_j, C_{j,k})$ and $W^Q(S_j, C_{j,k})$ as the impression allocation recommended by CMAB and RL strategy respectively. It locks all the impressions on the prediction date to compute the long-term reward for each of them using (2) with a sliding window of T days and averages them over the state-action pair to estimate $\bar{q}(S_j, C_{j,k})$. After that, as mentioned earlier, we use (4) and (5) to estimate T^R

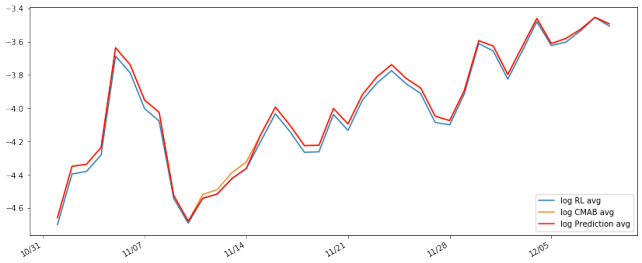

Figure 5: Time series plot of the daily prediction made by the evaluation framework for module A during Nov/Dec time window. In the graph, for each prediction date we have plotted logarithm of average cumulative long-term rewards T^R and T^Q for CMAB and RL. The blue curve represents expected total long-term rewards by RL model. The orange curve represents expected total long-term rewards by CMAB model. Optimal strategy prediction by the evaluation framework is shown by the red curve. The evaluation framework takes the expected cumulative long-term reward of the predicted optimal strategy. During this time window, the prediction curve often overlaps with T^R shown by orange curve. For this analysis the value of the parameter T is set as 7.

and T^Q as total rewards associated with the strategies CMAB and RL respectively and pick the model corresponding to the maximum expected cumulative long-term reward as the ground truth label.

Fig.4 shows the time series plot of the daily prediction made by the evaluation framework for module A during September/October time window. In the graph, for each prediction date we have plotted logarithm of average cumulative long-term rewards T^R and T^Q for CMAB and RL. Optimal strategy prediction by the evaluation framework is shown by the red curve. In our implementation, we make the evaluation framework take the expected cumulative long-term reward of the predicted optimal strategy. During this window, the prediction curve often overlaps with RL T^Q shown by orange curve. For this analysis the value of the parameter T is set as 7.

Fig.5 shows the time series plot of predictions made by the evaluation framework on Module A during November/December time window. The orange CMAB expected cumulative long-term reward T^R curve is often overlapped by the red evaluation framework prediction output.

3.3 Prediction Accuracy at different T values

The significance of T is to define how long in the future the model will consider in order to estimate the optimal long-term rewards. In our analysis we observed that different value of T may impact prediction accuracy. Table 1 shows the prediction accuracy of evaluation framework over the entire data set of 3 months on module A for different values of T.

During the course of our analysis, there were several changes to the underlying distribution of customer-content interactions space. To understand how changes in customer traffic and constantly changing content pool affect prediction accuracy, we divided the entire data set into 2 parts – September/October time window and November/December time window. For different values of T we analysed the prediction accuracy on the two data sets separately.

Table 1: Overall Prediction Accuracy for Module A

T value [*days*]	Accuracy [%]
5	80.0
7	67.9
10	71.7
12	69.2
14	60.3

Table 2 summarizes the prediction accuracy at different T values for the September/October window and November/December window separately. During this window, the expected cumulative long-term reward estimations for RL and CMAB are similar, as the model training was bootstrapped in early September. During this window, the reward estimations by both the models are somewhat noisy and unstable.

Table 2: Prediction Accuracy for different dataset windows for Module A

T value [*days*]	Sept/Oct Window [%]	Nov/Dec Window [%]
5	74.7	87.0
7	54.1	91.1
10	55.5	94.0
12	50.0	85.0
14	55.8	78.4

For the November/December time window, the prediction accuracy for module A is much more improved across every value of T. During this time, there was a significant increase in overall traffic, with 30% traffic contributed by new customers who were not mapped to any customer segment. As any prediction model performance depends on the richness of its training dataset, we believe this additional data is the reason for improved prediction accuracy in November/December time window.

3.4 RL predictability Analysis

Besides prediction accuracy, we wanted to understand how accurately the evaluation framework predicts RL strategy in a non-stationary customer-content space. The ground truth labels in the November/December dataset were almost all CMAB. This was contributed by three main factors that changed the underlying data distributions – overall increase in customer footprint across the website, with significant increase in the new customers traffic that belonged to no known context. Secondly, there was a drastic increase in the content pool across different contexts and lastly the evolving customers' behavior. Examining the customers activities at the context level, we observed the niche customer segments lost large portion of impressions, while there was a huge increase in customers activities towards few popular contexts. The average RL based reward estimations for these smaller customer segments

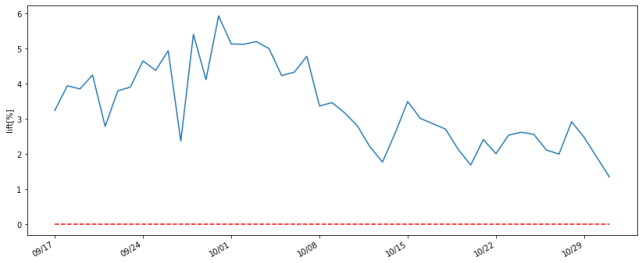

Figure 6: Time series plot of the lift calculated using $(T^Q - T^R)/T^R$ on module A for September/October time window. The blue curve shows the lift with respect to baseline CMAB model.

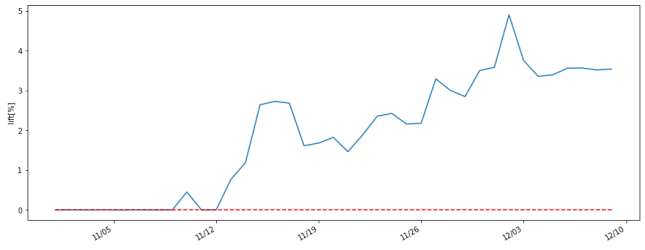

Figure 7: Time series plot of the lift calculated using $(T^R - T^Q)/T^Q$ on module A for November/December time window. The blue curve shows the lift with respect to baseline RL model.

grew significantly smaller in comparison to the estimations in September/October dataset.

For the September/October dataset, optimal policy labels and evaluation system predictions were skewed towards CMAB. We evaluated the prediction output as a RL classification problem and observed average precision as about 30% across all T values. On further examining the reward distributions and optimal policy at the individual context level, the average RL precision rate was above 70% across different T values . This led to a realization that there may exist few dominating contexts that may favor the optimal policy to maximize their reward estimations. Our website receives a large proportion of new customers as well as inactive customers who haven't visited in past few months. They together form the 'unknown' customer segment. This customer segment consistently favored CMAB. Hence we filtered them out to re-evaluate RL predictability.

Table 3 shows the precision recall evaluation for RL strategy predictions for the September/October dataset after filtering out the 'unknown' customer segment. The RL strategy prediction accuracy significantly improved.

Table 3: Precision Recall Analysis for RL prediction on Module A after filtering unknown context customers

T value [*days*]	Precision [%]	Recall [%]
5	89.33	1.0
7	84.9	1.0
10	87.81	1.0
12	88.43	1.0
14	87.02	1.0

3.5 Context level prediction of optimal reward strategy

At the module level, the evaluation framework uses a greedy approach to predict optimal reward strategy. It is driven by the popular customer contexts/states which have either higher expected rewards or larger customer footprint, or both. In our analysis we observed that when these prominent customer segments favored

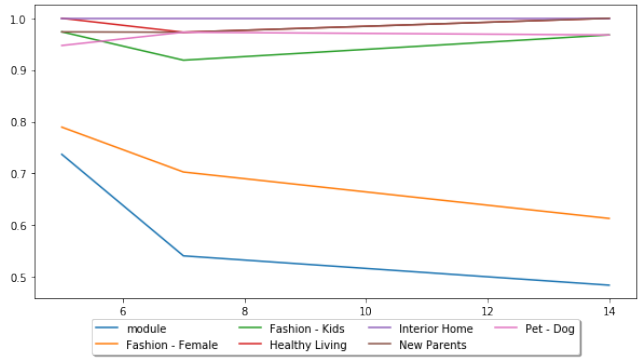

Figure 8: Plot of prediction accuracy for optimal policy prediction at the (global) module level compared to at the state/context level across different values of T

short-term reward strategy(CMAB), the evaluation framework (and the ground truth) also inclined towards them. In such scenarios, few customer segments/states influence the reward strategy for other smaller and niche customer segments/states. Overtime, this may negatively impact content discovery and exploration, leading to sub-optimal expected long-term reward. What if the optimal reward strategy for a recommendation module is decided at the customer's context or state instead for all customer contexts collectively. By doing that we make very personalized decision at the customer's current state and serve the most relevant recommendation using the optimal reward strategy. In our initial experiment to predict optimal reward strategy at the customer context/state, we have observed higher prediction accuracy than at the global level which we refer as module level. Fig.8 shows the graph with prediction accuracy for different customer contexts and the module level prediction accuracy.

4 CONCLUSION AND FUTURE WORK

In this paper, we presented a novel evaluation framework to automate the choice between RL and CMAB as the impression allocation strategy in an e-commerce setting where both customers' behaviors and content pool are assumed to be non-stationary. While one might be tempted to assume that long-term reward optimization

using RL is optimal, our results indicate that there are certain scenarios where CMAB is better suited to reap long-term returns. As is evident, an appeal of our model is its dynamic nature and the fact that it is fully automatic, completely driven by data. Our system is deployed in production in one of the top e-commerce portal, and is able to handle web scale traffic (millions of impressions on a daily basis) with significant impact to financial metrics.

The trade off between exploration and exploitation is generally complex, and a general solution for non-stationary environments is largely an open problem. RL algorithms learn reward associations with respect to their action space, and by nature they are well suited for stationary environments. Any decoupling or reversal from learnt associations takes longer and requires plenty of data. In our experience, the use of annealing parameter to discount rewards helped with learning new associations rapidly. Nevertheless, considering the dynamic nature of the environment as an input to the model is a ripe area for future research. Perhaps there has been studies that indicate that humans are willing to explore more when the amount of time left in a game or task is small(and not the earned reward at that point in time) with the hope that they will uncover to eventually exploit [5].

In the current implementation, the evaluation framework predicts the optimal strategy using the expected rewards across all underlying contexts. It is worth mentioning that while using CMAB and RL, in several cases we observed significant variance of rewards in the underlying bandits, there were a few contexts that offered higher rewards (with low noise estimates). If the optimization strategy is purely driven by the expectation over contexts, it is possible to end up with sub-par performance at the context level. An area ripe for further research is to select the strategy that is optimal for a context or more generally to a cohort of contexts. Additionally, it leads to a deeper question, what makes a good representation for a context, and can we learn rich high-dimensional representation of a context.

REFERENCES

[1] Shipra Agrawal and Navin Goyal. 2011. Analysis of Thompson Sampling for the multi-armed bandit problem. *CoRR* abs/1111.1797 (2011). arXiv:1111.1797 http://arxiv.org/abs/1111.1797

[2] Shipra Agrawal and Navin Goyal. 2012. Thompson Sampling for Contextual Bandits with Linear Payoffs. *CoRR* abs/1209.3352 (2012). arXiv:1209.3352 http://arxiv.org/abs/1209.3352

[3] Kamyar Azizzadenesheli, Emma Brunskill, and Animashree Anandkumar. 2018. Efficient Exploration through Bayesian Deep Q-Networks. *2018 Information Theory and Applications Workshop (ITA)* (2018).

[4] Richard S. Sutton; Andrew G. Barto. 1998. *Introduction to Reinforcement Learning*. MIT Press.

[5] Laura Carstensen, Derek Isaacowitz, and Susan Charles. 1999. Taking time seriously: A theory of socioemotional selectivity. *The American psychologist* 54 (04 1999), 165–81. https://doi.org/10.1037//0003-066X.54.3.165

[6] Olivier Chapelle and Lihong Li. 2011. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*. 2249–2257.

[7] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2010. A Contextual-Bandit Approach to Personalized News Article Recommendation. *CoRR* abs/1003.0146 (2010). arXiv:1003.0146 http://arxiv.org/abs/1003.0146

[8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *Arxiv preprint* arXiv:1312.5602 (2013).

[9] Brendan O'Donoghue, Ian Osband, Remi Munos, and Vlad Mnih. 2018. The Uncertainty Bellman Equation and Exploration. *Proceedings of the 35th International Conference on Machine Learning* (2018).

[10] Ian Osband, John Aslanides, and Albin Cassirer. 2018. Randomized Prior Functions for Deep Reinforcement Learning. *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (2018), 8626–8638.

[11] Ian Osband, Benjamin Van Roy, Daniel J. Russo, and Zheng Wen. 2017. Deep Exploration via Randomized Value Functions. *Arxiv preprint* arXiv:1703.07608 (2017).