Automatic Ad Format Selection via Contextual Bandits

Liang Tang
School of Computer Science
Florida International Univ.
11200 S.W. 8th St.
Miami, FL 33199
ltang002@cs.fiu.edu

Rómer Rosales
Applied Relevance Science
LinkedIn
2029 Stierlin Ct.
Mountain View, CA 94043
rrosales@linkedin.com

Ajit P. Singh
Applied Relevance Science
LinkedIn
2029 Stierlin Ct.
Mountain View, CA 94043
ajsingh@linkedin.com

Deepak Agarwal
Applied Relevance Science
LinkedIn
2029 Stierlin Ct.
Mountain View, CA 94043
dawargal@linkedin.com

ABSTRACT

Visual design plays an important role in online display advertising: changing the layout of an online ad can increase or decrease its effectiveness, measured in terms of click-through rate (CTR) or total revenue. The decision of which layout to use for an ad involves a trade-off: using a layout provides feedback about its effectiveness (exploration), but collecting that feedback requires sacrificing the immediate reward of using a layout we already know is effective (exploitation). To balance exploration with exploitation, we pose automatic layout selection as a contextual bandit problem. There are many bandit algorithms, each generating a policy which must be evaluated. It is impractical to test each policy on live traffic. However, we have found that offline replay (a.k.a. exploration scavenging) can be adapted to provide an accurate estimator for the performance of ad layout policies at Linkedin, using only historical data about the effectiveness of layouts. We describe the development of our offline replayer, and benchmark a number of common bandit algorithms.

Categories and Subject Descriptors

H.1.0 [**Information Systems**]: Models and PrinciplesGeneral

General Terms

Algorithms, Experimentation, Measurement

Keywords

Online advertising, personalization, recommender systems, layout, offline evaluation, bandit algorithms, exploration/exploitation

1. INTRODUCTION

Online advertising continues to grow rapidly as a fundamental component of the internet economy and ecosystem. It has been a subject of considerable research in machine learning, data mining, economics, user interfaces, and social sciences in general, among other areas. A widespread form of on-line advertising called *display advertising* consists of placing text or graphical ads (including a combination of these or related media) in certain reserved slots on publisher web-pages, targeted to specific user segments. Some common slots used for these purposes are the top and right panels of the page, but the location and layouts that could be employed are practically limitless.

Two common forms of payment methods used in display advertising include pay per impression (CPM) or pay per action. Typically actions consists of a click (CPC) or some post-click outcome like purchase, subscription, etc. referred to as conversion (CPA). In a CPM campaign, the advertiser pays when an ad is shown to the specified user segment at the specified location on a page, while for CPC the advertiser pays only if the user clicks on the ad[1]. Publishers typically sell display advertising using two methods — a) Guaranteed delivery where ad impressions are sold in advance and the volume to be delivered by the publisher is guaranteed to the advertiser. Since the publisher guarantees a certain number of impressions in the future using this method, advertisers typically pay a premium. Such methods are typically used by brand advertisers. b) Another increasingly popular mechanism for buying/selling advertising is to sell each impression through an auction with no guarantees on impression volume to advertisers. When a user requests a page with an ad space, an auction is run among the eligible ads (normally those that satisfy certain targeting constraints). Then, one or more ads, depending on the space and layout

[1]other performance variants like conversions can be treated as CPC by methods described in this paper

characteristics, are chosen for display, ranked based on some function of expected performance and advertiser bid. Advertisers are free to change their auction bids as they see fit to maximize their return on investment (ROI). Normally the location and layout of the available space (even within the same web-page) influence their bidding decisions. Such a method is typically used by performance advertisers. In this paper, we will only consider the auction method of selling display ads.

An online display advertisement is a combination of text, graphics, and potentially other media (e.g., photographs, videos), placed in a reserved space on a web page. We focus on the pay-per-click model, where payment is received when a user clicks on the advertisement. Each serve of an advertising spot on a web page to a user is referred to as an impression. An auction mechanism determines which ads are selected for an impression, and the price an advertiser will pay if their ad is clicked [9, 10]. A critical component of the auction is an estimate of the probability that an ad will be clicked on a given impression; also known as click-through rate (CTR) prediction [16, 1, 11, 6, 4, 15]. While there is a large literature on methods to predict CTR of ads, the concern of this paper is with optimizing the layout (also referred to as format) of an ad in addition to the choice of a high performing ad for an impression. The same ad content can be displayed in different ways: e.g., changing the background, border, fonts, the size of images, the degree/type of interactivity, the spacing of elements, the order of text and images, etc. We use the terms *layout* and *format* interchangeably, to refer to any difference in the look and feel of an advertising slot. Figure 1 contains four examples of ad layouts. While there are many different layouts, not all layouts are feasible for a given impression and set of ads: e.g., the layout might be too large or small; or a format might be specific to certain ad content, such as video ads.

The notion that the design of an ad affects ad performance predates online advertising [8]. However, such designs are often manually selected by humans. In this context, [7] presented an approach for modeling click response for different ad arrangement templates on a web-page. Our main **contribution** is to formalize the layout optimization problem as a contextual bandit problem (Section 2). The policy produced by the contextual bandit defines a distribution over layouts, which can be used to choose best performing layouts for ads on future impressions. We show such algorithmic format selection significantly improves performance relative to manual selection on the LinkedIn self-serve display advertising application. While we make no claim to new algorithms for generating contextual bandit policies, we evaluate the performance of several algorithms on data from Linkedin (Section 4) through *offline* policy evaluation. In our experience, it is important to have an accurate offline estimation method of whether a particular policy will work well in practice, without having to first deploy it. Simply testing many different policies on live traffic using A/B testing methods is costly. In fact, several bandit policies have tuning parameters that have to be selected carefully to ensure good performance, offline policy evaluation is an effective mechanism to do so. We share our experience with adapting an existing technique for policy evaluation, offline replay [14] (a.k.a. exploration scavenging [12]) to layout optimization (Section 3).

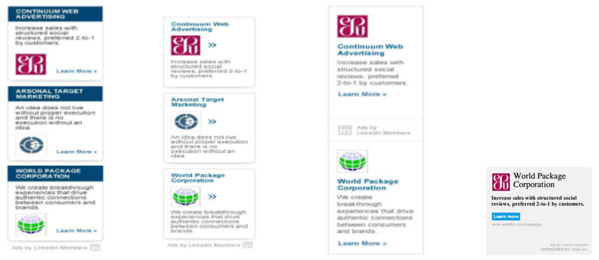
(a) Format 1, 160x600px (b) Format 2, 160x600px (c) Format 3, 160x600px (d) Format 4, 300x250px

Figure 1: Examples of ad layouts. Each arm is restricted to a certain size on a page.

2. LAYOUT OPTIMIZATION

We begin by posing ad layout optimization as an instance of a contextual multi-armed bandit problem [13]. Each of the K possible ad layouts is represented by an arm. The set of all arms (often referred as the action space) is denoted as $\mathcal{A} = \{a_1, \dots a_K\}$, where a_i denotes the i^{th} format/arm. Although the action space in our scenario is dynamic since new formats are introduced and old ones retired, we shall not consider this in our preliminary exposition to avoid notational clutter. When a page is requested by a user, the ad server is called to fill up all reserved ad slots. Each slot has a size constraint and an available set of ad formats to choose from. The ad-server has to choose one of the formats for the slot and the ads that will be shown within the format. We note that for the LinkedIn self-serve system, a given format may display more than one ad. Thus, a format may consist of more than one ad impression. A click on any ad within the format is considered a click on the format itself. Every instance to select an ad format on a page will be referred to as an *opportunity*.

In this paper, we decouple layout optimization from ad selection. We assume the ad selection algorithm is fixed and only focus on the problem of selecting the best format for a given request. While joint optimization of both ad-format and ad selection is desirable, it is more complex (especially due to tight latency constraints the ad-server has to adhere to), and left as future work.

2.1 Ad format selection and contextual multi-armed bandits

A multi-armed bandit problem is a sequential decision making process; e.g., the stream of page views that include ad slots is a sequence. Bandit problems involve making a decision in each round; e.g., deciding which layout $a \in \mathcal{A}$ to use for a given opportunity. Once a decision is made an observation is collected, and the corresponding reward computed; e.g., measuring whether the user clicked on the ad format, with the reward either being the click itself, or the payment for the click (pay-per-click).

An important extension of bandit problems is the addition of context or side information. The context for opportunity t is encoded as a feature vector $\mathbf{x}_t \in \mathcal{X}$. Context can include anything we know about opportunity t. Context can include features of a predictive model, as well has hard constraints: e.g., if the format has a 300 x 250 allocated pixel space, we cannot choose arms/layouts that are larger or smaller. Every layout is represented by an arm $a \in \mathcal{A}$, but context allows us to select a subset of feasible layouts for each opportunity t.

For any opportunity t, we can encode whether or not a user would have clicked on any of the ads displayed using layout $a \in \mathcal{A}$ as $\mathbf{r}_{t,a} \in \{0,1\}$. The vector of rewards for each layout/arm is denoted $\mathbf{r}_t = \{\mathbf{r}_{t,1}, \ldots, \mathbf{r}_{t,K}\}$. For theoretical exposition, we assume the reward vector is drawn from an (unknown) distribution that is i.i.d. We further denote the expected reward when arm a is displayed for opportunity t as $\bar{\mathbf{r}}_a$.

Each opportunity t of a contextual bandit can be decomposed into three steps:

1. The world draws a context and reward from an unknown distribution: $(\mathbf{x}_t, \mathbf{r}_t) \sim \mathcal{D}$. The context is revealed to the player; the reward is not.
2. The player uses some policy $\pi : \mathcal{X} \to \mathcal{A}$ to choose an arm $\pi(\mathbf{x}_t)$, given the revealed context.
3. The world reveals the reward of the choice, $\mathbf{r}_{t,\pi(\mathbf{x}_t)}$.
4. (Optional) The policy π is revised with the data collected for this opportunity, $\big(\mathbf{x}_t, \pi(\mathbf{x}_t), \mathbf{r}_{t,\pi(\mathbf{x}_t)}\big)$.

A policy is simply a function that maps the context to an arm. Policies can be deterministic or non-deterministic. For instance, a hard-coded manual policy to choose format for different ad slots by humans is a deterministic policy. A non-deterministic policy maintains a distribution over arms, $P(\pi(\mathbf{x}) = a \,|\, \mathbf{x})$, choosing one of them at runtime by sampling from the distribution. Layout optimization seeks to learn a policy which maximizes the average expected reward per opportunity

$$R(\pi) = \mathbb{E}_{(\mathbf{x})\sim\mathcal{D}}\left[\sum_{a\in\mathcal{A}} \bar{\mathbf{r}}_a P\left(\pi(\mathbf{x}) = a \,|\, \mathbf{x}\right)\right]. \qquad (1)$$

We have defined the reward as a click, so layout optimization is learning a policy which maximizes the click-through rate. If we redefine the reward as product of bid and click-throughs, the goal becomes maximizing expected revenue.

For the most part, this paper concerns itself with the setting where an opportunity consists of steps 1-3. A non-deterministic policy π is learned from data, prior to the first opportunity based on historical data, and does not change. The *online bandit* setting includes step 4, where the player adapts their policy for the next opportunity using data collected for the current one. To distinguish the two settings, we refer to an online policy as π_t, for $t = 1 \ldots T$. The online bandit problem does not require that the policy be updated after each t; in Section 3.1.2, the policy is updated periodically in a batched mode[2].

3. POLICY EVALUATION

The difficult part of maximizing Equation 1 is evaluating $R(\pi)$ on a given policy. Consider evaluation strategies used in other settings:

1. **A/B testing**: To compute the marginal expectation with respect to context distribution, segment the impressions or users into B buckets at random, and serve each bucket with a different policy and estimate the theoretical expectation with empirical average. Each bucket may need a substantial amount of traffic to generate a test with sufficient power In large online systems, there can be a substantial engineering cost in deploying many models. Not all policies are better than the current serving one; exploring the policy space often involves sacrificing immediate expected reward for higher expected future reward. Furthermore, optimal performance of a given policy often involves tweaking some constants, running A/B tests for this purpose is often prohibitive.

2. **Build a model for $\mathcal{D}$**: Computing $R(\pi)$ is straightforward if we have the probability density function for $\mathcal{D}$. However, this would generally involve building an accurate model of user activity, including when a user will click on an ad. Accurately modeling $\mathcal{D}$ would allow for solving computational advertising beyond just layout optimization. Modeling $\mathcal{D}$ is not very practical, since historical data does not typically contain observations on all arms show in a given context[3].

What we do have in historical tracking data is a log of i) past impressions, along with necessary context; ii) what format was used to display for an opportunity, i.e., the arm chosen; iii) whether the displayed format was clicked or not, i.e., the reward. The formats are chosen using a fixed serving policy $s : \mathcal{X} \to \mathcal{A}$, so the historical data consists of a time-ordered stream $\{(\mathbf{x}_t, s(\mathbf{x}_t), \mathbf{r}_{s(\mathbf{x}_t)})\}_{t=1}^{\tau}$, where τ is the present.

In the published literature on contextual bandits, using historical data from policy s to evaluate other policies $\pi \neq s$ is referred to as offline replay [14], or exploration scavenging [12]. Offline replay describes a class of sample estimators

$$\hat{R}(\pi) = \frac{1}{T}\sum_{t=1}^{T}\sum_{a\in\mathcal{A}} \mathbf{r}_{s(\mathbf{x}_t)} \mathbf{1}[\pi(\mathbf{x}_t) = s(\mathbf{x}_t)] w_{t,a}, \qquad (2)$$

where $\mathbf{1}[\cdot]$ is the indicator function, and $w_{t,a}$ is a normalization weight,

$$w_{t,a} = \frac{1}{P(s(\mathbf{x}_t) = a \,|\, \pi(\mathbf{x}_t) = a)}.$$

In our ad layout selection problem, the contextual features of an opportunity are the channel (web page) and the layout size. Each channel and size combination has its own set of admissible formats. In other words, a layout can only be recommended for one channel and one layout size. Let c_t, ℓ_t denote the channel and layout size of impression $\mathbf{x}_t$, $t = 1, ..., T$. Then,

$$P(\pi(\mathbf{x}_t) = a) = P(\pi(\mathbf{x}_t) = a \,|\, c_t, \ell_t).$$

For any other channel c or size ℓ, $c \neq c_t$ or $\ell \neq \ell_t$,

$$P(\pi(\mathbf{x}_t) = a \,|\, c, \ell) = 0.$$

[2]In the online bandit setting, the order of the rounds matters, which is not reflected in Equation 1. The sample estimator for $R(\pi)$, Equation 2, is readily adapted to the online setting.

[3]a.k.a. the partial-label problem [14].

Let $r_t = \mathbf{r}_{s(\mathbf{x}_t)} \mathbf{1}[\pi(\mathbf{x}_t) = s(\mathbf{x}_t)]$. By substituting $w_{t,a}$ and r_t into Eq. 2, we have

$$\begin{aligned}
\hat{R}(\pi) &= \frac{1}{T}\sum_{t=1}^{T}\sum_{a\in\mathcal{A}} \frac{r_t}{P(s(\mathbf{x}_t) = a \,|\, \pi(\mathbf{x}_t) = a)} \\
&= \frac{1}{T}\sum_{t=1}^{T}\sum_{a\in\mathcal{A}} \frac{r_t \cdot P(\pi(\mathbf{x}_t) = a)}{P(s(\mathbf{x}_t) = a \,, \pi(\mathbf{x}_t) = a)} \\
&= \frac{1}{T}\sum_{t=1}^{T}\sum_{a\in\mathcal{A}} \frac{r_t \cdot P(\pi(\mathbf{x}_t) = a \,|\, c_t, \ell_t) \cdot P(c_t, \ell_t)}{P(s(\mathbf{x}_t) = a \,, \pi(\mathbf{x}_t) = a)} \\
&= \frac{1}{T}\sum_{c,\ell} P(c,\ell) \cdot \sum_{t=1}^{T}\sum_{a\in\mathcal{A}} \frac{r_t \cdot P(\pi(\mathbf{x}_t) = a \,|\, c, \ell)}{P(s(\mathbf{x}_t) = a \,, \pi(\mathbf{x}_t) = a)}
\end{aligned}$$

Given a channel c and a layout size ℓ, $P(\pi(\mathbf{x}_t) = a | c, \ell)$ only depends on a. Hence, we can let $h_{c,\ell,a} = P(\pi(\mathbf{x}_t) = a \,|\, c, \ell)$ and substitute it into $\hat{R}(\pi)$, we have

$$\begin{aligned}
\hat{R}(\pi) &= \frac{1}{T}\sum_{c,\ell} P(c,\ell) \cdot \sum_{t=1}^{T}\sum_{a\in\mathcal{A}} \frac{r_t \cdot h_{c,\ell,a}}{P(s(\mathbf{x}_t) = a \,, \pi(\mathbf{x}_t) = a)} \\
&= \sum_{c,\ell} P(c,\ell) \cdot \sum_{a\in\mathcal{A}}\sum_{t=1}^{T} \frac{r_t \cdot h_{c,\ell,a}}{T \cdot P(s(\mathbf{x}_t) = a \,, \pi(\mathbf{x}_t) = a)} \\
&\approx \sum_{c,\ell} P(c,\ell) \cdot \sum_{a\in\mathcal{A}} \frac{R_{c,\ell} \cdot h_{c,\ell,a}}{M_a}, \qquad (3)
\end{aligned}$$

where $R_{c,s}$ is the total reward for policy π obtained within channel c and layout size s, M_a is the number of matched impressions for arm a. Eq. 3 requires estimating various distributions since they are unknown. We let:

$$\begin{aligned}
R_{c,\ell} &= \sum_{t\in\{t \,|\, c_t = c, \ell_t = \ell\}} r_t, \\
h_{c,\ell,a} &\approx \frac{|\{t \,|\, \pi(\mathbf{x}_t) = a, c_t = c, \ell_t = \ell\}|}{|\{t \,|\, c_t = c, \ell_t = \ell\}|}, \\
P(c,\ell) &\approx \frac{|\{t \,|\, c_t = c, \ell_t = \ell\}|}{T}, \\
M_a &= T \cdot P(s(\mathbf{x}_t) = a \,, \pi(\mathbf{x}_t) = a) \\
&\approx |\{t | s(\mathbf{x}_t) = a, \pi(\mathbf{x}_t) = a\}|.
\end{aligned}$$

As can be seen, the accuracy of the estimate $\hat{R}(\pi)$ depends in turn on how well the various probabilities can be estimated. Since this is done from historical data, this data plays a major role in guaranteeing a good estimate. For example, if the serving policy s rarely serves some layouts a, this could lead to a high variance estimate. Similarly, if the evaluated policy π tends to serve some a very few times the variance could increase although this could be a consequence of a providing very little reward. In an extreme case, if s never serves a, there is no way to know the reward of π if π always recommends a. This is to a large extent equivalent to the problem that arises in importance sampling when the proposal distribution has near zero probability in areas the distribution of interest has non-negligible mass.

3.1 Implementation in Hadoop

In order to evaluate the replayer estimation framework discussed in the previous section at scale, we implemented it in the Map/Reduce framework. The testing event set is partitioned into several subsets. Each reducer only handles the evaluation for one event subset. The general data flow is shown in Figure 2.

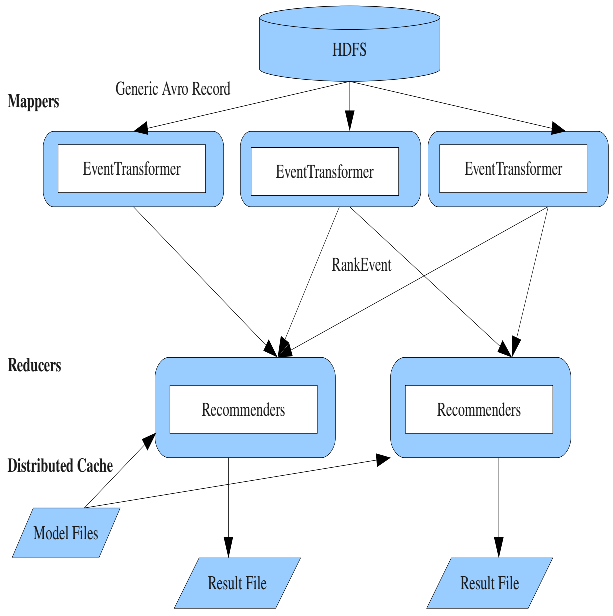

Figure 2: Data Flow In Hadoop

In Figure 2, the historical events (such as opportunities) are stored in the distributed file system (HDFS). The mappers transform every event record into a ranked event by using a specified event transformer. The ranked event is a unified data structure consisting of event attributes, time stamp and reward. Testing policies are implemented as recommender instances in every reducer. The replayer in the reducers feed every ranked event to each recommender and records their output.

We considered two scenarios for offline evaluation. One scenario for static recommendation policies, where the models never change during the evaluation, but are kept fixed from the start of the evaluation. In this scenario, all models are built before the evaluation starts. The other scenario is the case of dynamic recommendation policies, where the models are updated based on the user feedback recorded from the events that they recommended. The static scenario omits Step 4 (Section 2.1); the dynamic scenario includes it.

Based on these two scenarios, we consider two different methods to shuffle and partition the ranked events from the mappers.

3.1.1 Without Feedback

If the recommendation policies are static and there is no feedback from the test events, the ranked events from the mappers can be randomly assigned to any reducer. The replayer only needs to address the correct data size balance for the reducers. In our implementation, the Hadoop partitioner is computed by a hash function taking as argument the time stamp of the event.

3.1.2 With Feedback

If the recommendation policies are dynamic and the replayer needs to provide feedback to these policies, the ranked events from the mappers cannot be randomly shuffled and assigned because the sequential order of the test events af-

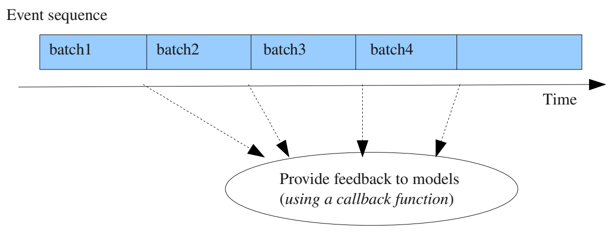

Figure 3: Batched Feedback

fects the test policies. For some bandit algorithms, such as the ϵ-first algorithm, the exploration is done only on the events at the beginning so we cannot break the original order of the historical events. To keep the original event order, the time stamp of the ranked event is a part of the key given to the reducers. As a result, Hadoop is able to sort all rank events by their time stamps before sending the reducers.

In a real system, the user feedback cannot in general be retrieved immediately after each impression. In each reducer, the event sequence is split into a collection of batches. The feedback is accumulated within a batch and only provided to the recommendation models at the end of each batch (see Figure 3). In our implementation the batch size is a free parameter, given by the user.

4. EVALUATION

In this section, we first present the performance of various recommendation policies for ad format selection by using the introduced replayer. Then, we compare the accuracy of the replayer estimation with the live (online) system for some policies. All of the experiments are in the context of the LinkedIn advertising platform using data from certain randomly selected markets. Adequate measures were taken to preserve user privacy while conducting all analyses.

Table 1: Recommendation Policies Employed

Algorithm	Description
ϵ-greedy(ϵ)	With probability ϵ it recommends an item uniformly at random, otherwise it recommends the best item (arm) [17].
ϵ-ngreedy(c,d)	A variant of ϵ-greedy algorithm, where ϵ decreases with time. Parameters c and d control the speed of deceasing [3].
ϵ-first(ϵ)	Recommends an item uniformly at random for the first ϵ proportion of time and always recommends the best item for the remaining test events [18].
softmax(α, k)	Randomly draws an item from the multinomial item distribution with parameters $\theta = (\theta_1, ...\theta_m)$ to recommend, where $\theta_i = \frac{k+P_i^\alpha}{\sum_{j=1}^m (k+P_j^\alpha)}$, P_i is the average reward for the i-th item, and m is the number of items [17].
softmax(α)	softmax(α, 0). Note that softmax(0) is the uniform random policy, which is used as a baseline policy.
ϵ-softmax(ϵ, α, k)	With probability ϵ it recommends an item uniformly at random, otherwise recommends according to softmax(α, k).
ϵ-firstsoftmax(ϵ, α)	Recommends an item uniformly at random for the first ϵ proportion of time and uses softmax(α) to recommend for the remaining test events.
ucb(ρ)	A variant of UCB1 (Upper Confidence Bound) algorithm with exploration parameter ρ [3].
Thompson sampling	Beta-Bernoulli Thompson Sampling [5].
original	Original serving policy, employed to generate the historical data.

4.1 Ad Format Selection

The goal in the ad format selection problem considered in this section is to select appropriate ad formats that maximize the cumulative pay-off: the cumulative CTR (click-through-rate) or alternatively revenue. Other performance quantities are possible but we will use these as they are simple to measure for the purposes of this paper. As discussed before, in real scenarios new formats are added into the online system on a regular basis. The recommendation models can better predict the popularity of the new formats as they are displayed. Thus, the trade off between exploration and exploitation is a fundamental element in this problem. Therefore, this problem is formalized as a bandit problem, where an arm corresponds to an ad format (or an *item* in general) and the reward is a click (binary reward) or revenue (continuous value reward), depending on the problem.

4.1.1 Comparison of Bandit Algorithms

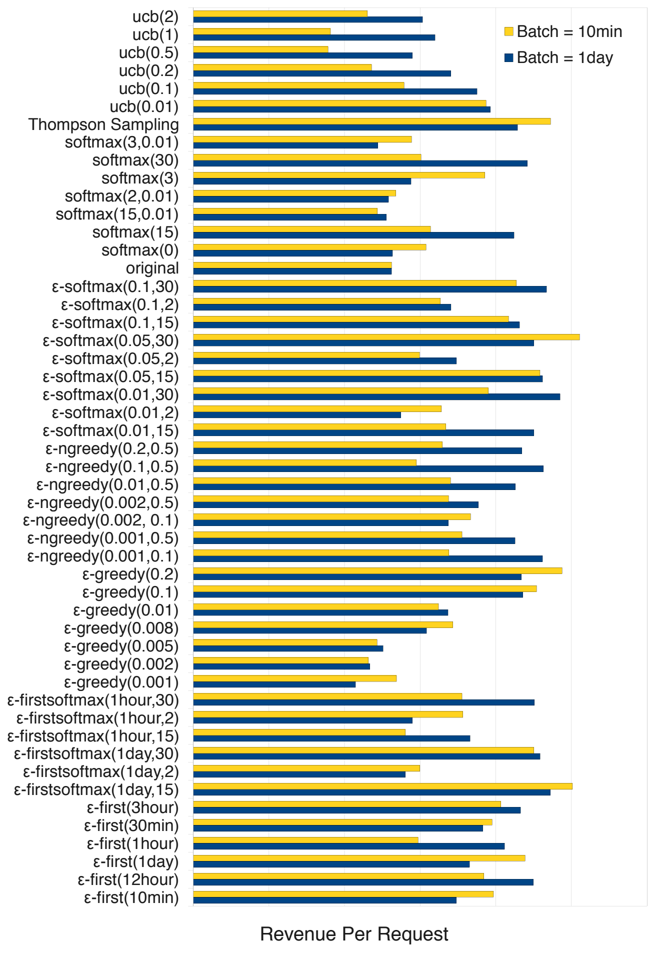

Figure 4: Comparison of bandit algorithms without historical training data

In this evaluation, we consider two general experimental settings. In the first setting, used for evaluating traditional multi-armed bandit algorithms, policies do not have any historical information about the ad formats. This setting is useful to evaluate the effect of new ad formats. However, as most of the ad formats are not new at any given time, we

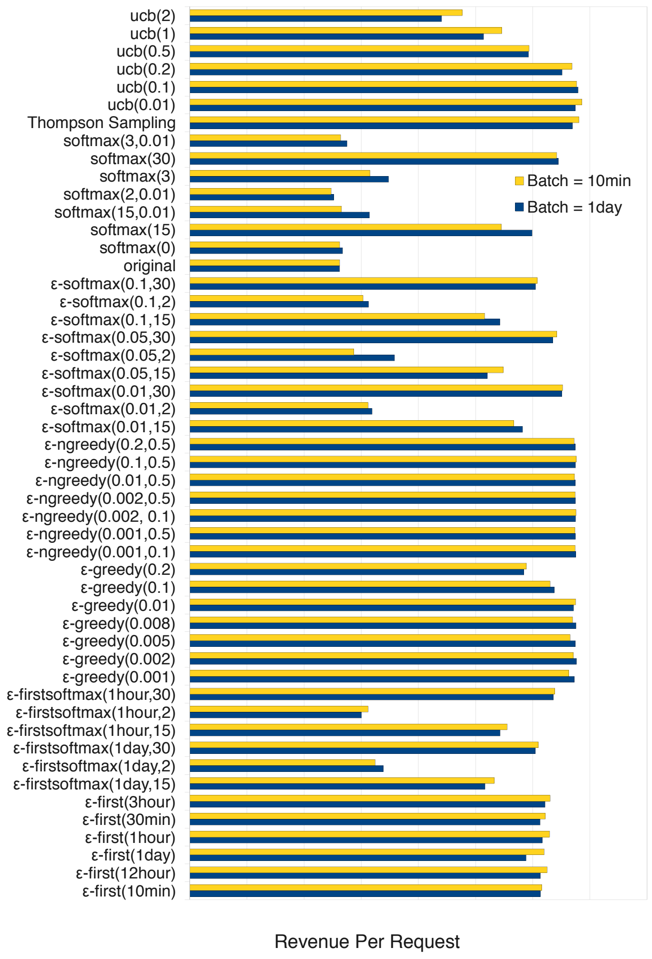

Figure 5: Comparison of bandit algorithm with 7 days of historical training data

also employ another experimental setting to look at these policies when we have historical data.

The testing data consists of about 184M sampled impression events collected for a period of several weeks. There are more than 40 channels and each channel has about 2 to 15 layout sizes. The ad formats are not uniformly served in the historical data because some channels receive a large amount of traffic and some a relatively smaller amount. Since some policies are randomized algorithms and sensitive to the order of the testing events, we randomly split the entire data sequence into 5 subsequences and run every policy independently. The results are then averaged appropriately using the number of matched events (Eq. 3). Table 1 summarizes the multi-armed bandit algorithms used in this evaluation. Note that softmax(0) is the uniform random policy. In this evaluation, softmax(0) is regarded as a baseline policy.

Figure 4 shows the revenue per request when no historical data is available at the beginning of the experiment (cold-start setting). Figure 5 shows the revenue per request with the previous week of historical data available for training (warm-start). The x-axis has undergone scaling to remove confidential information. Batch = t indicates that a time of length t is employed to simulate updating the recommendation models. We use either one day or 10 minutes, this means that the system will update the recommendation models daily or every then minutes. In addition to this, we also vary different parameters for the bandit algorithms considered, as indicated in the graph.

As shown in Figure 4, by choosing appropriate parameters, ϵ-softmax, ϵ-firstsoftmax and ϵ-greedy have the best performances. For ϵ-softmax and ϵ-firstsoftmax, when α is large, their strategies are close to ϵ-greedy. ϵ-ngreedy is a variant of ϵ-greedy [3] with ϵ decreasing as follows:

$$\epsilon = \min\{1, \frac{cK}{d^2 n}\},$$

where c and d are parameters, K is the number of formats that can be recommended, and n is the total number of recommendations.

There are many variations of the UCB algorithm. In this experiment, we apply a variant of the UCB1 algorithm, UCB1(ρ), which can be seen as a generalized UCB1 algorithm [2]. Given an ad format request, assume there are K formats that can be recommended, the UCB1(ρ) algorithm recommends the format:

$$\arg\max_{i\in\{1,\ldots,K\}} \frac{r_i}{T_i} + \sqrt{\frac{\rho \log n}{T_i}},$$
$$n = \sum_i^K T_i,$$

where r_i is the total revenue from the i-th format, T_i is the total number of recommendations for the i-th format, and ρ is a parameter controlling the priority of exploration versus exploitation. When ρ is larger there is more exploration. In the experiment, ρ is varied from 2 to 0.01. When $\rho = 2$, UCB1(ρ) is the original UCB1 algorithm. In Figures 4 and 5, UCB1(0.01) achieves higher revenue than any other UCB1(ρ). This means that, for this particular setting, we do not need to spend a lot of events on exploration and should focus more on exploitation. In Figure 4, for "Batch = 10 min", UCB(ρ) with $\rho >= 0.1$ is even worse than the uniform random policy (softmax(0)). This is because the UCB prefers to explore the format that has a large uncertainty. However, those formats may be underperforming formats (whose CTRs are lower than the average CTR). Therefore, the UCB policy could be worse than the random policy.

In Figures 4 and 5, the results for ϵ-softmax, ϵ-firstsoftmax, ϵ-greedy and UCB1(ρ) are similar to the conclusion mentioned in [3]. Through parameter tuning the performance of ϵ-greedy and UCB1(ρ) can approach the optimal one. When the algorithms are not well-tuned they can degrade rapidly. However, the optimal parameter values depend on the distribution of the data set. If the data distribution changes, the tuned parameters may not be as effective anymore. On the other hand, if we already have historical data for parameter tuning, those multi-arm bandit algorithms are in a warm-start setting rather than cold-start setting. The best parameter tuned in one setting may not be the best for the other setting. As shown in Figure 4 (cold-start), ϵ-greedy with larger ϵ performs well. But in Figure 5 (warm-start), there are 7 days' historical training data, so ϵ-greedy with smaller ϵ performs better. Note that the best ϵ-greedy in Figure 5 is better than the best ϵ-greedy in Figure 4.

For ϵ-ngreedy, [3] asserts that it should approach the optimal performance by careful parameter tuning. However, [19] also indicates that the family of ϵ-decreasing algorithms does not have a clear advantage over ϵ-greedy. Our evaluation results shown in Figures 4 and 5 confirm the point in [19].

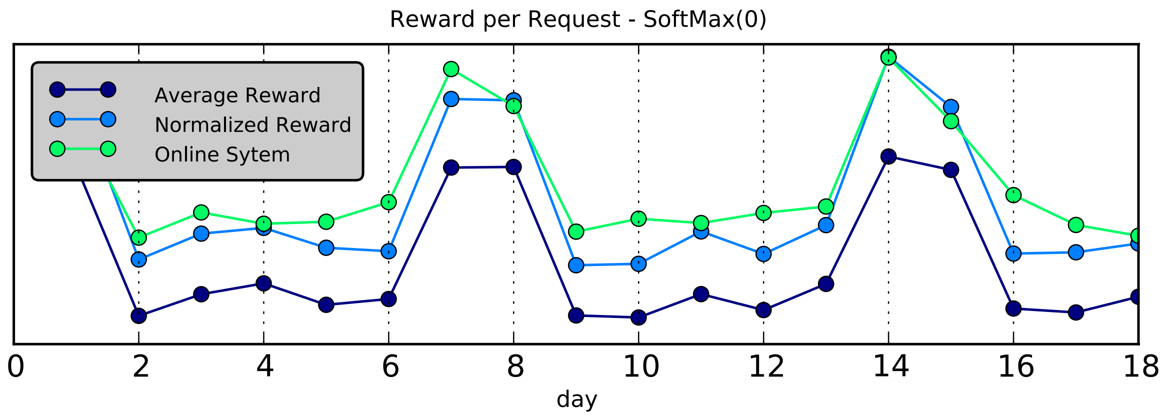

Figure 6: Daily Revenue Per Request for softmax(0)

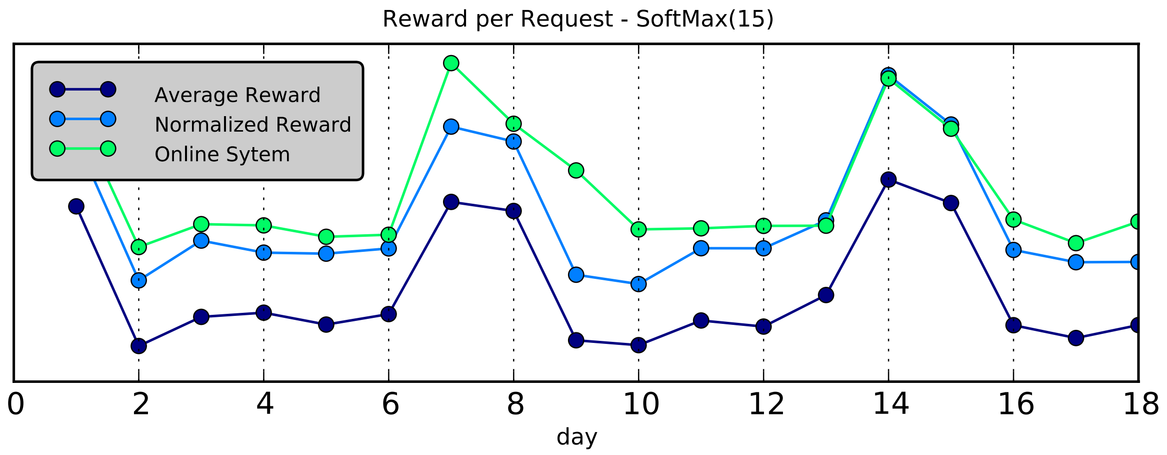

Figure 7: Daily Revenue Per Request for softmax(15)

The performance of Bernoulli Thompson sampling is close to the best observed performance when the batch size is 10 minutes. However, when we increase the batch size to 1 day, it cannot adjust its Beta distributions in time. As a result, it wastes some impressions on exploration and spends less on exploitation. Considering the cold-start and warm-start settings together, Bernoulli Thompson sampling is overall the best policy in our evaluation for ad format selection.

4.2 Accuracy of Replayer Estimation

In order to evaluate the accuracy of the replayer estimates described, we use data from the online LinkedIn ad system, where various policies were implemented, as the source of ground-truth. In the online production systems, the traffic is split into several portions that run different recommendation policies. For this paper we select the traffic where two of these policies are run. In particular, we selected two extreme policies for evaluation: softmax(0) and softmax(15). We calculate daily revenue per request for the online system and let this be our ground-truth performance for these policies. We use the historical data to run the replayer's offline evaluation approach and estimate the revenue per request for the same policies.

In Figures 6 and 7 we can observe the difference between the estimated value and the actual value for the metric of interest. This shows the accuracy of the replayer estimates. The average reward is computed as the total reward over the number of matched events, as in [14]. The historical data is not uniformly served since the ad recommendation has to be optimized. As a result, the average reward estimate is biased [14]. As shown by these figures, the average reward always underestimates the performances of softmax(0) and softmax(15). The normalized reward is the estimate proposed in Section 3. As shown by the two figures, the normalized reward is always closer to the actual value of the revenue per request than the average reward.

5. CONCLUSION

This paper presented a study of optimizing real-time, webpage ad layout selection, where the goal is to optimize user response. Optimal ad layout selection was formulated as an instance of the contextual bandit problem. As evaluating a battery of bandit algorithms on an on-line system is costly and impractical, the paper described a method for large scale offline evaluation. In particular, we have formulated a method that can be used even in cases where the serving policy is biased (*i.e.,* items are not served at random). This is an important practical problem for which a formal approach has not been provided in detail in the past. In addition we provided a practical system design taking advantage of the Hadoop-MapReduce architecture. We provided an experimental comparison among many bandit algorithms in the context of a large system, the LinkedIn Ad platform. Additionally, we compared the proposed offline policy evaluation approach with the on-line production system and demonstrate its accuracy.

We did not consider the interaction between ad-format selection and ad selection within a format. A complete solution would require joint optimization that makes the problem considerably difficult since the number of ads in a system is typically large. Further, ranking at runtime has to be done under strict latency constraints. Thus, evaluating the goodness of all possible format and ad combinations is not feasible, approximate procedures are needed. We plan to pursue such an approach in the future.

6. REFERENCES

[1] D. Agarwal, B.-C. Chen, and P. Elango. Spatio-temporal models for estimating click-through rate. In *WWW*, pages 21–30, 2009.

[2] J.-Y. Audibert, R. Munos, and C. Szepesvári. Exploration-exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19):1876–1902, 2009.

[3] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[4] K. Bauman, A. Kornetova, V. Topinskii, and D. Khakimova. Optimization of click-through rate prediction in the Yandex search engine. *Automatic Documentation and Mathematical Linguistics*, 47(2):52–58, 2013.

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

[6] O. Chapelle, E. Manavoglu, and R. Rosales. Simple and scalable response prediction for display advertising. *Transactions on Intelligent Systems and Technology*, (to appear), 2013.

[7] H. Cheng, E. Manavoglu, Y. Cui, R. Zhang, and J. Mao. Dynamic ad layout revenue optimization for display advertising. In *Workshop on Data Mining for Online Advertising*, 2012.

[8] D. S. Diamond. A quantitative approach to magazine advertisement format selection. *Journal of Marketing Research*, 5(4):376–386, Nov 1968.

[9] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242–259, 2005.

[10] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 99(2):430–434, 2009.

[11] T. Graepel, J. Q. Candela, T. Borchert, and R. Herbrich. Web-scale Bayesian click-through rate prediction for sponsored search advertising in Microsoft's Bing search engine. In *ICML*, pages 13–20, 2010.

[12] J. Langford, A. L. Strehl, and J. Wortman. Exploration scavenging. In *ICML*, pages 528–535, 2008.

[13] J. Langford and T. Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in neural information processing systems*, pages 817–824, 2007.

[14] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In I. King, W. Nejdl, and H. Li, editors, *WSDM*, pages 297–306. ACM, 2011.

[15] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, S. Chikkerur, D. Liu, M. Wattenberg, A. M. Hrafnkelsson, T. Boulos, and J. Kubica. Ad click prediction: a view from the trenches. In *KDD*, 2013.

[16] M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: estimating the click-through rate for new ads. In *WWW*, pages 521–530, 2007.

[17] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 9(5):1054–1054, 1998.

[18] L. Tran-Thanh, A. C. Chapman, E. M. de Cote, A. Rogers, and N. R. Jennings. Epsilon-first policies for budget-limited multi-armed bandits. In *AAAI*, 2010.

[19] J. Vermorel and M. Mohri. Multi-armed bandit algorithms and empirical evaluation. In *ECML*, pages 437–448, 2005.