Learning to Rank in the Position Based Model with Bandit Feedback

Beyza Ermis, Patrick Ernst, Yannik Stein, Giovanni Zappella
{ermibeyz,peernst,syannik,zappella}@amazon.de
Amazon
Berlin, Germany

ABSTRACT

Personalization is a crucial aspect of many online experiences. In particular, content ranking is often a key component in delivering sophisticated personalization results. Commonly, supervised learning-to-rank methods are applied, which suffer from bias introduced during data collection by production systems in charge of producing the ranking. To compensate for this problem, we leverage *contextual multi-armed bandits*. We propose novel extensions of two well-known algorithms viz. LinUCB and Linear Thompson Sampling to the ranking use-case. To account for the biases in a production environment, we employ the *position-based click model*. Finally, we show the validity of the proposed algorithms by conducting extensive offline experiments on synthetic datasets as well as customer facing online A/B experiments.

KEYWORDS

Multi-armed bandits, Position-based model, Content Ranking

ACM Reference Format:
Beyza Ermis, Patrick Ernst, Yannik Stein, Giovanni Zappella. 2020. Learning to Rank in the Position Based Model with Bandit Feedback. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3340531.3412723

1 INTRODUCTION

The content catalogue in many online experiences today is too large to be disseminated by regular customers. To explore and consume these catalogues, content providers often present a selected subset of their content which is personalized for easier consumption. For example, almost all major music streaming services rely on vertical tile interfaces, where the user interface is subdivided into rectangular blocks, vertically and horizontally. The content of every tile is a graphical banner. Usually, customers observe a limited number of tiles, that sometimes even rotate every few seconds, where only one large banner is visible at each point in time.

The selected tiles displayed to the customer significantly impact the engagement with the service. Moreover, the order in which they are presented by the application strongly impacts their chance of being observed by the customer. This clearly calls for the need to consider the order as well as the bias introduced by the visualization mechanism. Generally, the selection and ranking of content are core operations in most modern recommendation and personalization systems. In this problem setting, we need to leverage all available information to improve the customer experience.

Related Work. Learning-to-rank approaches have been studied in practical settings (e.g., see [10]) and there is additional work to address the presence of incomplete feedback (also known as "bandit" feedback) (e.g., [16–18, 26]). Learning-to-rank can be cast as a combinatorial learning problem where, given a set of actions, the learner has to select the ordered subset maximizing its reward. A standard combinatorial problem with bandit feedback (e.g., see [3, 6]) would provide a single feedback (e.g., click/no-click signal) for each subset of selected actions or tiles, making the problem unnecessarily difficult. A more benign formulation is to look at the problem as a semi-bandit problem, where the learner can observe feedback for each action, eventually transformed by a function of the actions position in the ranking. Recently, several relevant methods have been proposed for this kind of problem: non-contextual bandit methods such as [16, 18, 19, 21] *do not leverage side-information* about customers or content and thus do not present a viable solution for our problem setting. Different approaches offer solutions using complex click models (i.e., the cascade model [17, 27]), which can be effective on applications like re-ranking of search results, but are complex to extend to consider other aspects like additional elements on the page since in practice they are often controlled by different subsystems. The approaches described in [8, 14, 22] share the same problem space as this work, but target different aspects of the problem, such as fairness, reward models, and evaluations.

Contribution. The first contribution of this paper is two different contextual linear bandit methods for the so called *Position-Based Model* (PBM) [5], which are straightforward to implement, maintain and debug. Second, we provide an empirical study on techniques to estimate the position bias during the learning process.

Specifically, we introduce new algorithms derived from LinUCB and linear Thompson Sampling with Gaussian posterior, addressing the problem of learning with bandit feedback in the PBM. This model assumes that the probability of the customer interacting with a piece of content is a function of the relevance of that content and the probability that the customer will actually inspect that content allowing the model to be used in various scenarios. To the best of our knowledge, this is the first contextual bandit approach using PBM. Finally, we show the validity and versatility of our approach by conducting extensive experiments on experiments on synthetic datasets as well as customer facing online A/B experiments, including lessons learned with anecdotal results.

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3412723

2 PROBLEM SETUP

In the following we introduce the Position-Based Model (PBM) to distinguish rewards for different ranking positions and afterwards the linear reward learning model.

Position-Based Model. PBM [7, 23] is a click model where the probability of getting a click on an action depends on both its relevance and position. In this setting, each position is randomly observed by the user with some probability. It is parameterized by both L action dependent relevance scores, expressing the probability that an action is judged as relevant, and L position dependent examination probabilities $q \in [0,1]^L$, where q_ℓ denotes the examination probability that position ℓ was observed (also known as position bias). The core assumption of PBM is that the observed click Bernoulli variable C depends on two other hidden Bernoulli variables E and R where R represents the relevance of an action a to a context x and E represents the event whether a user examines an action a at a certain position ℓ: $P(C=1|x,a,\ell) = P(E=1|\ell)P(R=1|x,a)$. In Section 4, we discuss how we derive q_ℓ.

The Learning Model. We consider a linear bandit setting in which the taken action at each round is a list of L actions chosen from a given set $\{a_1,\dots,a_K\}$ of size K. Accordingly, assuming a semi-bandit feedback, we receive a reward in the form of a list of feedbacks corresponding to each position of the recommended list. At each round t of the learning process, we obtain K vectors in $\mathbb{R}^d$ that represent the available actions for the learner. We denote these by $\mathcal{A}_t = \{a_t^1,\dots,a_t^K\}$ and the action list selected at time t will be denoted as $A_t = (A_t^1,\dots,A_t^L)$, where A_t is a permutation of $\mathcal{A}_t$.

The PBM is characterized by examination parameters $(q_\ell)_{1\le\ell\le L}$, where q_ℓ is the probability that the user effectively observes the item in position ℓ. At round t, the selection A_t is shown and the learner observes the complete feedback. However, the observation Z_t^ℓ at position ℓ is censored being the product of the examination variable Y_t^ℓ and the actual user feedback C_t^ℓ where $Y_t^\ell \sim \mathcal{B}(q_\ell)$ and $C_t^\ell = {A_t^\ell}^T\theta + \eta_t^\ell$ with all η_t^ℓ being 1-subgaussian independent random variables. When the user considered the item in position ℓ, Y_t^ℓ is unknown to the learner and C_t^ℓ is the reward of the item shown in position ℓ. Then, we can compute the expected payoff of each action in each position, conditionally on the action: $\mathbb{E}[Z_t^\ell|A_t^\ell] = q_\ell {A_t^\ell}^T\theta$, where $\theta \in \mathbb{R}^d$ is the unknown model parameter. At each step t, the learner is asked to make a list of L actions A_t that may depend on the history of observations and actions taken. As a consequence to this choice, the learner is rewarded with $r_{A_t} = \sum_{\ell=1}^L Z_t^\ell$, where $Z_t = (Z_t^1,\dots,Z_t^L) = (C_t^1 Y_t^1,\dots,C_t^L Y_t^L)$. The goal of the learner is to maximize the total reward $\sum_{t=1}^T r_{A_t}$ accumulated over the course of T rounds.

3 RANKING ALGORITHMS

We now introduce two contextual bandit algorithms for learning to rank in the PBM. The first one is named *LinUCB-PBMRank* that is a variation of LinUCB [1, 4, 9], the contextual version of the optimistic approaches inspired by UCB1. The second algorithm, called *LinTS-PBMRank*, is Bayesian approach to exploration and it is a variation of linear Thompson Sampling (LinTS) [2].

3.1 The Optimistic Approach: LinUCB-PBMRank

The LinUCB algorithm for contextual bandit problem for a single action case at each time t, obtains a least square estimator for θ using all past observations: $\hat{\theta}_t = \arg\min_{\tilde{\theta}\in\mathbb{R}^d} \sum_{s=1}^{t-1}(C_s - {A_s}^T\tilde{\theta})^2 + \lambda\|\tilde{\theta}\|^2$. We can now derive a conditionally unbiased estimator of the model parameter θ for the ranking case in the PBM as a least square solution of $\hat{\theta}_t = \arg\min \sum_{s=1}^{t-1}\sum_{\ell=1}^L (Z_s^\ell - q_\ell {A_s^\ell}^T\tilde{\theta})^2 + \lambda\|\tilde{\theta}\|^2$.

Proposition 1. *The solution to the convex optimization problem formulated above gives a closed form solution for the estimator $\hat{\theta}$:*

$$\hat{\theta}_t = V_t^{-1} b_t = \left(\sum_{\ell=1}^L q_\ell^2 V_t^\ell + \lambda I\right)^{-1}\left(\sum_{\ell=1}^L q_\ell b_t^\ell\right) \tag{1}$$

where $\forall\,\ell \in [L], V_t^\ell = \sum_{s=1}^{t-1} A_s^\ell {A_s^\ell}^T$ *and* $b_t^\ell = \sum_{s=1}^{t-1} Z_s^\ell A_s^\ell$.

At each round t, concentration inequalities (see [1]) provide a confidence ellipsoid around $\hat{\theta}_t$ that contains θ with high probability. The confidence ellipsoid that controls the reliability of the estimator given a fixed sequence of vectors of actions $A_1,\dots A_{t-1}$ in $\mathbb{R}^d$, for any unit vector $x \in \mathbb{R}^d$, with probability $1-\delta$ is:

$$\left\langle x, \hat{\theta} - \theta \right\rangle \le \|x\|_{V_t^{-1}}^2 \sqrt{f_{t,\delta}} \quad \text{with} \qquad f_{t,\delta} = 2\log(1/\delta)$$

The pseudocode of LinUCB for ranking in the PBM is given in Algorithm 1.

3.2 The Bayesian Approach: LinTS-PBMRank

From a Bayesian point of view, the problem can be formulated as a posterior estimation of the parameter θ. Here, the true observations Z_t^ℓ is replaced by its conditional expectation given the censored position variables $Y_t^\ell \sim \mathcal{B}(q_\ell)$. We introduce the filtration $\mathcal{F}_t$ as the union of history until time $t-1$, and the contexts at time t, $\mathcal{F}_t = (A_1, Z_1, \dots, A_t)$ such that for all t, ℓ, $\mathbb{E}[Z_t^\ell|\mathcal{F}_t] = \mathbb{E}[C_t^\ell|\mathcal{F}_t]\mathbb{E}[Y_t^\ell|\mathcal{F}_t] = q_\ell({A_t^\ell}^T\theta)$. We present a fully Bayesian treatment of Linear Thompson Sampling where we assume σ^2 follows an Inverse-Gamma distribution and θ follows a multivariate Gaussian:

$$\sigma^2 \sim \mathcal{IG}(\alpha_0, \beta_0) := p_0(\sigma^2)$$
$$\theta \sim \mathcal{N}(\theta_0, \sigma^2 V_0^{-1}) := p_0(\theta)$$
$$Z_t^\ell | A_t^\ell, \theta, q_\ell, \sigma^2 \sim \mathcal{N}(q_\ell \theta^T A_t^\ell, \sigma^2)$$

For the above model, the joint model posterior $p(\theta, \sigma^2|\mathcal{F}_t)$ follows a Normal-Inverse-Gamma distribution. We can compute the posterior of the full-Bayesian approach as follows:

$$p(\tilde{\theta}|\mathcal{F}_t) \propto p_0(\sigma^2)p_0(\theta)\prod_{t=1}^T\prod_{\ell=1}^L p(Z_t^\ell|\theta_t, \mathcal{F}_t)$$
$$\propto \exp\{-\frac{1}{2\sigma^2}\sum_{\ell=1}^L (Z_t^\ell - q_\ell {A_t^\ell}^T\theta_t)^T(Z_t^\ell - q_\ell {A_t^\ell}^T\theta_t)\}$$
$$\exp\{-\frac{1}{2\sigma^2}(\theta_t-\theta_0)^T V_0^{-1}(\theta_t-\theta_0)\}\{(\sigma^2)^{-(\alpha_0+1)}\exp\left(-\frac{\beta_0}{\sigma^2}\right)\}$$

We rearrange the posterior to formalize the posterior mean θ_t and the variance V_t^{-1} in closed form. First, we rewrite the quadratic

Algorithm 1 LinUCBPBMRank

Input: Position Bias Parameters $(q_1, \dots, q_L)$, confidence level $\delta > 0$, regularization λ.
for $t = 1, \dots, T$ **do**
Get the contextualized actions $\mathcal{A}_t$,
Compute $\hat{\theta}_t$ as in Prop. 1 and for all $a \in \mathcal{A}_t$,

$$U_t(a) = a^T \hat{\theta}_t + \sqrt{f_{t,\delta} \|a\|^2_{V^{-1}}}$$

Build Top-L action list

$$A_t \in \arg\max_{a \in \mathcal{A}_t} \sum_{\ell} q_\ell U_t(a)$$

(ties broken arbitrarily)
Update $V_t \leftarrow V_{t-1} + \sum_\ell q_\ell^2 A_t^\ell A_t^{\ell T}$
Receive feedback for round t
Update $b_t \leftarrow b_{t-1} + \sum_\ell q_\ell Y_t^\ell A_t^\ell$

Algorithm 2 LinTSPBMRank

Input: Position Bias Parameters $(q_1, \dots, q_L)$, prior precision parameters α_0 and β_0, so that
$\sigma^2 \sim \mathcal{IG}(\alpha_0, \beta_0)$ and $p_0(\theta) = \mathcal{N}(0, \sigma I)$.
for $t = 1, \dots, T$ **do**
Get the contextualized actions $\mathcal{A}_t$,
Sample $\tilde{\theta}_t \sim p_{t-1}$
Compute scores for all

$$a \in \mathcal{A}_t : s_t(a) = a^T \tilde{\theta}_t$$

Build Top-L action list,

$$A_t \in \arg\max_{a \in \mathcal{A}_t} \sum_{\ell} q_\ell s_t(a)$$

Update $V_t \leftarrow V_{t-1} + \sum_\ell q_\ell^2 A_t^\ell A_t^{\ell T}$
Receive feedback for round t
Update $b_t \leftarrow b_{t-1} + \sum_\ell q_\ell Y_t^\ell A_t^\ell$

terms in the exponential as a quadratic form:

$$\begin{aligned} Q(\tilde{\theta}, \sigma^2) &= (Z_t^\ell - q_\ell A_t^{\ell T} \theta_t)^T (Z_t^\ell - q_\ell A_t^{\ell T} \theta_t) + (\theta_t - \theta_0)^T V_0^{-1} (\theta_t - \theta_0) \\ &= (\tilde{Z}_t^\ell - W\theta_t)^T (\tilde{Z}_t^\ell - W\theta_t) \end{aligned}$$

where

$$\tilde{Z}_t^\ell = \begin{pmatrix} Z_t^\ell \\ V_0^{\frac{1}{2}} \theta_0 \end{pmatrix} \quad \text{and} \quad W = \begin{pmatrix} q_\ell A_t^\ell \\ V_0^{\frac{1}{2}} \end{pmatrix}$$

In this case $V_t^{-1} = (W^T W)^{-1} = (q_\ell^2 A_t^{\ell T} A_t^\ell + V_0^{-1})^{-1}$ and $\theta_t = \Sigma_t (W^T \tilde{Z}_t^\ell) = V_t^{-1}(q_\ell A_t^{\ell T} \tilde{Z}_t^\ell + V_0^{-1}\theta_0)$. At each time t, we sample one vector from the posterior for each action to compute the scores. The parameters of this posterior in terms of the parameters at time $t-1$ are analytically computed as:

$$V_t = (\sum_t q_\ell^2 A_t^\ell A_t^{\ell T} + V_0) \qquad \theta_t = V_t^{-1} b_t$$

$$\alpha_t = \alpha_0 + \frac{t}{2} \qquad \beta_t = \beta_0 + \frac{1}{2}(\eta_t - \theta_t^t b_t)$$

where

$$V_0 = \lambda I \qquad \eta_t = \eta_{t-1} + \sum_\ell Z_t^{\ell 2}$$

$$b_t = b_{t-1} + q_\ell Z_t^\ell A_t^\ell \qquad z_t = V_{t-1}^{-1} q_\ell A_t^\ell$$

We can simply apply the Sherman-Morrison identity [24] that computes the inverse of the sum of an invertible matrix as the outer product of vectors to improve computational efficiency. The linear Thompson Sampling to rank (LinUCB-PBMRank) is summarized in Algorithm 2. For dense action vectors the above update schema is computed in $O(d^2)$.

4 POSITION BIAS ESTIMATION

Accurate estimation of the position bias is crucial for unbiased learning-to-rank from implicit click data. We can provide these parameters either as fixed or use an automatic parameter estimation method. Using fixed hyperparameters in a production environment with many different use-cases and continuously expanding use cases can be quite challenging in terms of maintenance and scaling. To avoid that, we evaluate three automatic estimation methods: *i)* estimate using the click-through rate (CTR) per position by updating them online after observing each record *ii)* a supervised learning approach leveraging the Bayesian Probit regression (PR) model and *iii)* bias estimation using an expectation-maximization (EM) algorithm.

4.1 CTR per position

One of the most commonly used quantities in click log studies is click-through rates (CTR) at different positions [5, 15]. A common heuristic used in these cases is the rank-based CTR model where the click probability depends on the rank of the document $P(C = 1|\ell) = \rho_\ell$. Given the click event is always observed, ρ_ℓ can be estimated using MLE. The likelihood for the parameter ρ_ℓ can be written as:

$$\mathcal{L}(\rho_\ell) = \prod_{c_i \in S_c} \rho_\ell^{c_i} (1 - \rho_\ell)^{1 - c_i} \tag{2}$$

where S_c is the set of clicks and c_i is the value of the click of the i^{th} occurrence for position ℓ. By taking the log of (2), calculating its derivative and equating it to zero, we get the MLE estimation of the parameter ρ_ℓ. In this case, it is the sample mean of c_i's:

$$\rho_\ell = \frac{\sum_{c_i \in S_c} c_i}{| S_c |} \tag{3}$$

4.2 Probit Regression Model

The CTR-based method is very intuitive but does not consider actions' features and their probability of being clicked. Furthermore, it can incur in the same bias-related problem of the naive rankers since the clicks will likely be more frequent towards the beginning of the ranking. We aim to learn a mapping $x \rightarrow [0, 1]$ from a set of features x to the probability of a click. Bayesian Linear Probit model is a generalized linear model (GLM) with a Probit link function. The sampling distribution is given by: $P(C|\theta, x) := \Phi(C \cdot \theta^T x / \beta)$, where we assumed that C is either 1 (click) or 0 (no click) and Φ is the cumulative density function of the standard normal distribution: $\Phi(t) := \int_{-\infty}^{t} \mathcal{N}(s; 0, 1) ds$. It serves as the link function that maps the output of the linear model (sometimes referred to as the score) in $[-\infty, \infty]$ to a probability distribution in $[0, 1]$ over the observed data, C. The parameter β scales the steepness of the inverse link function. The function $P(C|\theta, x)$ is called likelihood as a function of θ and sampling distribution as a function of C; the latter is the

generative model of the data and is a proper probability distribution whereas the former is the weighting that the data, C, gives to each parameter. The model uncertainty over the weight vector is captured in $P(\theta) = \mathcal{N}(\mu, \sigma^2)$. Given a feature vector x, the proposed sampling distribution together with the belief distribution results in a Gaussian distribution over the latent score. Given the sampling distribution $P(C|\theta, x)$ and the prior $P(\theta)$, the posterior is calculated: $P(\theta|C, x) := P(C|\theta, x)P(\theta)$.
We keep a Probit Regression (PR) model for each position ℓ. Given the likelihood $P(C|x, a, \ell, \theta)$, the posterior is calculated as:
$P(\theta|C, x, a, \ell) := P(C|x, a, \ell, \theta)P(\theta)$.
Then, the predictive distribution $P(C|x, a, \ell)$ can be computed with given feature vector and the posterior (See [13] for details). As we mentioned in Section 2, the probability of getting a click on action a in position ℓ is equal to $P(C = 1|x, a, \ell) = P(E = 1|\ell)P(R = 1|x, a)$. Here, our goal is to compute $q_\ell = P(E = 1|\ell)$ and we compute it as:

$$q_\ell = \frac{P(E = 1|\ell)P(R = 1|x, a)}{P(E = 1|\ell = 1)P(R = 1|x, a)}$$

where we assume $P(E = 1|\ell = 1) = 1$. It has to be noted that although this assumption holds for the applications considered in the experimental section of this paper, this is not guaranteed in all real-world applications. It is possible that the content on the page may get reshuffled by another system and the position of the component visualizing the ranking changed, with significant impact on the chance for the customer to observe the content. In Section 4.3, we will provide experimental results where this assumption is violated.

4.3 Expectation-Maximization

After the observations made for the CTR and PR estimators, we explore different directions in order to provide a solution that can be more robust in real-world scenarios. The Expectation-Maximization (EM) algorithm can be applied to a large family of estimation problems with latent variables. In particular, suppose we have a training set $X = \{x_1, \ldots, x_n\}$ consisting of n independent examples. We wish to fit the parameters of a model $P(X, Z)$ to the data, where the likelihood is given by:

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \log P(X; \theta) = \log \sum_{Z} P(X, Z; \theta) \tag{4}$$

However, explicitly finding the maximum likelihood estimates of the parameters θ may be hard. Here, the $z^{(i)}$'s are the unobserved latent random variables. In such a setting, the EM algorithm gives an efficient method for maximum likelihood estimation. To maximize $\mathcal{L}(\theta)$, EM construct a lower-bound on $\mathcal{L}$ (E-step), and then optimize that lower-bound (M-step) repeatedly. The EM estimator provided in this section can be seen as a generalization of PR estimator, which should provide better practical performance. Given the relevance estimate $\gamma_{x,a} = P(R = 1|x, a)$, the position bias $q_l = P(E = 1|l)$ where $P(C = 1|x, a, l) = P(E = 1|l)P(R = 1|x, a)$, and a regular click log $\mathcal{L} = \{(c, x, a, \ell)\}$, the log likelihood of generating this data is:

$$\log P(\mathcal{L}) = \sum_{(c,x,a,\ell)\in\mathcal{L}} c \log q_\ell \gamma_{x,a} + (1 - c) \log(1 - q_\ell \gamma_{x,a}) \tag{5}$$

The EM algorithm can find the parameters that maximize the log-likelihood of the whole data. In [25], the authors introduced an EM-based method to estimate the position bias from regular production clicks. The standard EM algorithm iterates over the Expectation and Maximization steps to update the position bias q_ℓ and the relevance parameter $\gamma_{x,a}$. In this paper, we modify the standard EM and take $\gamma_{x,a}$ equal to $\sigma(A_t^T \hat{\theta}_t)$ at each step t where A_t is the contextualized action. In this way, we take the context information into account. At iteration $t + 1$, the Expectation step estimates the distribution of hidden variable E and R given parameters from iteration t and the observed data in $\mathcal{L}$:

$$P(E = 1|c, x, a, \ell) = \frac{(1 - \gamma_{x,a}^{(t)}) q_\ell^{(t)}}{1 - q_\ell^{(t)} \gamma_{x,a}^{(t)}} \tag{6}$$

Fore more details we refer to [25]. The Maximization step updates the parameters using the quantities from the Expectation step:

$$q_\ell^{(t+1)} = \frac{1}{T} \sum_{t} \left(c_\ell^{(t)} + (1 - c_\ell^{(t)}) \frac{(1 - \gamma_{x,a}^{(t)}) q_\ell^{(t)}}{1 - q_\ell^{(t)} \gamma_{x,a}^{(t)}} \right) \tag{7}$$

5 EXPERIMENTS

In this section we provide a number of empirical results to demonstrate the advantages of the proposed algorithms compared to their "naive" counterparts and other baselines. To this aim, we perform experiments on synthetic datasets with different variants of the position bias estimation in a controlled environment showing differences that would not be possible to measure in an online environment. We also tested our algorithms in online experiments against other baselines. In order to avoid the risk of negative customer experience, we ran online experiments comparing our two variants of the algorithms and two "safe" production-like baselines. Comparing our algorithms to baselines that provided negative results in the offline experiments «««< HEAD would be irresponsible and completely against the customers' interest. ======= would be irresponsible and completely against customers' interest.
»»»> 0cf90d8767ab6104db13b47590e1c46c96662bd9

5.1 Offline Experiments

For the purpose of testing our algorithms in a controlled environment, we created two synthetic datasets with 25 available actions and we limited the algorithm to select a maximum of 20 actions, simulating the behaviour of a page that does not display all the actions to all the customers. The actions vectors were generated as in the following: we fixed the number of dimensions to 5 and then generated dense vectors of random numbers in [0,1) and then set all the entries having a value below 0.1 to 0.0 (introduces some sparsity). The context vectors, part of the same datasets, are set to have 10 dimensions and generated in the same way of the actions. A simplified version of the behaviour of the production system is reproduced in the offline experiments, so we join the action and context vectors to create a *contextualized action*. This is created by concatenating the action vector, the context vector and the vectorized outer product of the two. This process generates 25 vectors, each one representing an action and each vector is made of three blocks: the action vector, the context vector, and the cross product between the action vector and the context vector. After the vectors are generated, they get normalized by dividing them by their respective square norms. The only vectors received by the predictors are the ones made available at the end of this process.
For the dataset with **real valued rewards** (later called **SINREAL**), the rewards are generated as follows: at the beginning of the process

	Number of actions/positions			
ALGORITHMS	1	5	10	20
LinUCB	48278.30 ± 5.99	69334.81 ± 144.02	68294.01 ± 171.0	65185.37 ± 415.50
LinUCB-PBMRank (Real)	48274.81 ± 3.51	75800.33 ± 5.58	76296.08 ± 6.53	76307.25 ± 7.86
LinUCB-PBMRank (EM)	48266.04 ± 10.63	74119.72 ± 219.15	74364.61 ± 133.72	74567.43 ± 273.93
LinUCB-PBMRank (PR)	48276.28 ± 3.51	75786.47 ± 22.45	76291.33 ± 6.01	76102.86 ± 40.49
LinUCB-PBMRank (CTR)	48247.31 ± 5.90	72104.12 ± 78.13	72981.72 ± 132.31	73610.40 ± 350.70
LinTS	48518.36 ± 12.93	67234.41 ± 209.95	69363.75 ± 327.70	68054.39 ± 124.57
LinTS-PBMRank (Real)	48559.73 ± 9.27	76194.49 ± 1.95	76709.01 ± 2.45	76803.95 ± 11.74
LinTS-PBMRank (EM)	48153.47 ± 49.21	74992.00 ± 91.78	75365.20 ± 167.04	76018.56 ± 96.39
LinTS-PBMRank (PR)	48559.73 ± 9.27	73690.61 ± 49.48	74670.29 ± 55.35	75940.56 ± 245.67
LinTS-PBMRank (CTR)	48523.96 ± 6.31	74254.12 ± 23.55	74682.50 ± 186.22	73128.87 ± 312.81
Random Selection	45044.14 ± 11.60	70780.10 ± 20.29	71253.58 ± 21.74	71273.89 ± 92.59

Table 1: Cumulative reward on SINREAL for different combination of algorithms and position bias estimation techniques. The name of the selected position bias estimator is reported in parenthesis and (Real) means no estimation if performed, but the real values used to generate the bias are provided to the algorithm.

	Number of actions/positions			
ALGORITHMS	1	5	10	20
LinUCB	34790.07 ± 5.55	49453.96 ± 430.82	50450.33 ± 65.66	20915.29 ± 2165.77
LinUCB-PBMRank (Real)	34682.12 ± 31.15	53081.15 ± 148.92	53436.85 ± 19.28	53538.65 ± 344.03
LinUCB-PBMRank (EM)	34568.03 ± 173.82	50562.10 ± 113.04	50069.72 ± 203.47	51397.31 ± 221.55
LinUCB-PBMRank (PR)	34584.33 ± 24.68	53178.26 ± 139.11	53352.61 ± 148.24	53470.40 ± 168.91
LinUCB-PBMRank (CTR)	34552.33 ± 124.94	50029.49 ± 132.16	50570.73 ± 225.96	49521.61 ± 302.92
LinTS	34850.33 ± 130.12	46402.29 ± 83.20	47707.79 ± 90.08	39598.70 ± 1228.26
LinTS-PBMRank (Real)	34882.33 ± 117.58	53700.02 ± 62.98	53945.51 ± 73.62	53939.03 ± 133.72
LinTS-PBMRank (EM)	34795.66 ± 113.52	47921.58 ± 26.23	48176.78 ± 235.97	51994.30 ± 103.47
LinTS-PBMRank (PR)	34882.33 ± 35.12	51877.87 ± 53.50	52283.41 ± 126.40	52444.39 ± 122.09
LinTS-PBMRank (CTR)	34766.01 ± 171.48	47657.17 ± 58.34	46218.11 ± 189.21	45815.60 ± 256.34
Random Selection	26721.66 ± 35.42	42139.62 ± 89.18	42389.31 ± 126.45	42552.10 ± 85.47

Table 2: Cumulative reward on SINBIN for different combination of algorithms and position bias estimation techniques. The name of the selected position bias estimator is reported in parenthesis and (Real) means no estimation if performed, but the real values used to generate the bias are provided to the algorithm.

	ALGORITHMS	5 / SINBIN	5 / SINREAL	10 / SINBIN	10 / SINREAL
ϵ=0.1	LinTS-PBMRank (Real)	48348.93 ± 61.61	68579.71 ± 14.53	48549.76 ± 87.88	69033.46 ± 43.39
	LinTS-PBMRank (EM)	46175.04 ± 150.30	67962.91 ± 56.57	46574.36 ± 178.47	68027.64 ± 158.46
	LinTS-PBMRank (PR)	47885.84 ± 234.51	66564.25 ± 50.06	47987.43 ± 267.57	67149.27 ± 302.14
ϵ=0.25	LinTS-PBMRank (Real)	39538.48 ± 103.30	57147.24 ± 32.55	39787.77 ± 183.72	57528.76 ± 111.05
	LinTS-PBMRank (EM)	38164.60 ± 380.26	55250.56 ± 232.94	38649.44 ± 384.07	55949.08 ± 234.69
	LinTS-PBMRank (PR)	37821.41 ± 504.89	52039.96 ± 398.44	37838.06 ± 534.24	53135.97 ± 397.89
ϵ=0.5	LinTS-PBMRank (Real)	25546.38 ± 109.99	38095.41 ± 43.26	25811.65 ± 197.66	38332.70 ± 50.64
	LinTS-PBMRank (EM)	25051.16 ± 128.68	37153.18 ± 382.06	25336.41 ± 540.04	37637.62 ± 127.40
	LinTS-PBMRank (PR)	23901.48 ± 408.30	34382.20 ± 313.71	23958.86 ± 708.51	34689.39 ± 432.71

Table 3: Cumulative reward on SINBIN when position bias is $\frac{1-\epsilon}{\exp(position)}$. The name of the bias estimator is reported in parenthesis. (Real) means no estimation is performed, but the real values used to generate the bias are provided to the algorithm.

a unit length random vector w is fixed and w will be used to compute the inner product with the contextualized actions, following the linear assumption made in Section 2. The reward is generated by summing the inner product between w and the contextualized action vector with a noise factor uniformly sampled in the interval [-0.1,0.1). Then, we apply floor and ceiling operations to make sure to obtain a reward in [0,1]. In the case of the dataset with **binary valued rewards** (later called **SINBIN**), the same procedure is followed but we binarize the rewards by thresholding them with a predefined hyperparameter. Before providing the rewards to update the predictor, the rewards are divided by the exponential of the position assigned to the corresponding action by the learning algorithm (this is done "online" and depends on the predictions made by the algorithm). The aim is to mimic the behaviour observed in online experiments with recommendation systems where the users tend to click significantly more on the top positions on the ranking. The exponential function was chosen after observing the behaviour of customers in some online experiments.

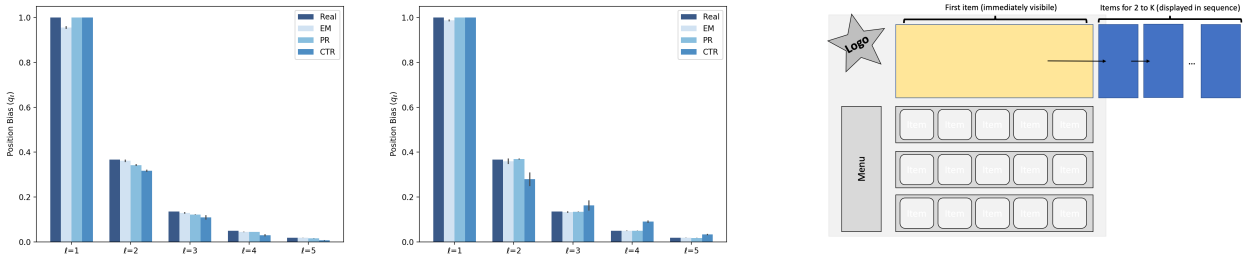


Figure 1: Comparison of the real position biases and the position biases estimated by CTR, PR and EM methods for the top 5 positions. SINBIN on the left and SINREAL on the middle. On the right: Structure of the page where the experiment was ran. The central-top slot is optimized using the methods described in this paper. All the items in the list get eventually displayed on the page since the slot is automatically loading the next piece of content after a fixed amount of time.

5.1.1 Results. In our experiments, we compared the two algorithms presented in Section 3.1 with their counterparts that do not account for the bias introduced by the ranking position namely LinTS and LinUCB. These algorithms select the actions taking the top-K with the highest scores instead of the single best one as in their original definition. The update operation is performed using all the selected actions and the corresponding rewards without any re-weighting. This is equivalent to set all the $\{q_\ell\}_{\ell=1}^{L}$ to 1 in the algorithms referenced above.

Synthetic Data Results. Tables 1 and 2 report the results of experiments run on synthetic data in order to validate our ideas in a controlled environment. The dataset used in this section are SINREAL and SINBIN, whose details are available in Section 5.1. Please note that the since the datasets are generated artificially, every potential prediction of the algorithm can receive the correct reward and we do not need to employ techniques for running offline evaluation with biased datasets (e.g., [20]). In these offline experiments, we can observe two important trends: i) not addressing the position bias can significantly mislead algorithms to the point that they can become worse than a random selection, ii) using an automatic method for estimating the position bias gives a clear advantage but there is no clear winner between PR and EM.

Position Bias Estimation Results. The previous experiments show that CTR is inferior as position bias estimation method, while PR and EM perform almost equally. In Figure 1 we compare the quality of the estimation methods by comparing the estimated position biases with the true values observed in the synthetic datasets. However, it is important to recall that for the CTR and PR estimators the parameter for the first position is artificially set to 1, while the EM method is performing its estimation without any additional information. This is particularly useful in cases where the hyperparameter associated with the first position is unknown because it is controlled by external factors (e.g., the ranked content is displayed in a position where does not catch the attention of the users). We conducted a range of experiments, reported in Table 3, to assess the sensitivity of PR with respect to this parameter. The results clearly show that the more severe the violation of the $q_1 = 1$ becomes, the better EM becomes compared to PR. This may seem a small details, but it is important in practice since recommendations are often served in web-pages containing several elements which may attract a significant amount of user attention before the user even gets to visualize the recommendations.

5.2 Online A/B Experimentation

To validate our offline results and to show the effectiveness of our approach in a real-world scenario, we conducted two end-customer facing online A/B tests. Due to the costs and potential negative customer experience of running A/B tests involving real paying customers, we focused on two main scenarios. In each scenario we pick one widget, a so-called carousel (Fig. 1c)[1] UI and is embedded at the top of the landing page of a large music streaming service.

We alter the arrangement of the items between control A and treatment B to test different baselines against configurations of our bandit-based ranking approach. Particularly we test:

- a human-curated list arrangement in control against our approach with fixed position biases, i.e. without online automatic estimation
- a collaborative filtering based ranking in control against our approach with online EM position bias estimation

The customers are split equally, 50%/50% random allocation, between control and treatment.

5.2.1 Online Learning to Rank A/B Test vs. Human-curated content. In this experiment the goal is to have a confined test for the bandit learning to rank algorithm and thus we purposefully do not include automatic position bias estimation. Instead, we rely on manual hyper parameters based on view events, where the parameter for position i is based on the number of historical customer requests that viewed i divided by the number of requests. The candidate widget consists of 50 candidate items, which were represented by banners containing music spanning different genres and user tastes (e.g., audio books, music for children). Our control treatment always shows the same order of 13 manually curated items to customers. In treatment, we apply our ranking bandit to contextually re-rank the candidate set every time the customer visits the landing page. We pick the top-13 scored banners to fill the carousel and present them to the customer. An action corresponds to the display of one such banner represented as a feature vector. In particular, to contextualize the ranking, we leverage different types of features representing the customer, content, and general context, such as temporal information, customer taste profiles and customer affinities towards musical content. These representations are used as input for the bandit, which is solving the LTR problem. We see

[1] A carousel consists of a list of banners, where only one banner is displayed at a time and rotated to the next one after a certain time period.

major increases of various classical ranking measurement and engagement metrics in treatment which leverages the ranking bandit. Overall, here customers interacted more with the widget and also consumed more music. In particular, if we compare the performance of the widget with the version provided to the control group, we are able to improve the following widget specific metrics: i) the mean reciprocal rank (MRR) increased by 15.38%, ii) the amount of attributed playbacks increased by 17.16%, iii) the listening duration measured in log seconds increased by 16.90% and iv) the number of customers playing music increased by 15.62%. All results are statistical significant with p-values below 0.001. These positive results are also found in the performance of the landing page which contains the widget: i) there is 2.33% more playbacks originating from the landing page, ii) the listening duration measured in log seconds increased over all customers increased by 3.04% and iii) the number of customers who played music increased by 2.81%. All results are statistical significant with p-values below 0.001. We also tracked the position of each banner in the carousel during the course of this experiment, which revealed that the approach is able to respond to intra-day short-term trends. For instance, specific types of contents are popular only at some time of the day and the algorithm is able to learn that. Such a case is depicted in Figure 2a, which plots the average ranks over time. As we can see, there is specific content which is popular during night time and the ranking bandit is able to capture intra-day trends thanks to temporal features provided as part of the context and fast model updates. Additionally, we observed that the ranking bandit was able to handle seasonal content: an example is shown in Figure 2b, which shows the average position of a banner targeted to Father's day (celebrated on May 30th) in Germany that was ranked high in the days leading to the holiday.

5.2.2 Online Learning to Rank A/B Test vs. Matrix factorization baseline. To validate that the gains of our algorithm are not only due to increased variety of displayed content (i.e. randomization effects) compared to the static version in the human baseline, we further test the ranking bandit with position bias estimation against a matrix factorization baseline. This was performed on the same carousel widget during summer 2019 over 8 days in the US. The carousel contained 10–15 banners that were manually curated and changed during the experiment. In the control group, the banners were ordered by scores derived from an existing production system that is based on matrix factorization. In the treatment, we applied the ranking bandit with position bias estimation. To contextualize, we used temporal features, as well as several features to represent the customer such as the customer's taste profile and the scores from the matrix factorization baseline.

We saw increased customer engagement in treatment compared to control, in particular along the following metrics for the targeted widget: i) the mean reciprocal rank (MRR) increased by 5.08%, ii) the attributed playbacks increased by 7.57%, iii) the listening duration measured in log seconds increased by 7.23% and iv) the number of customers playing music from this widget increased by 6.72%. All results are statistical significant with p-values below 0.001. We also improved metric for the whole landing page that contains the widget: i) the number attributed playbacks increased by 0.8% with p-value=0.075, ii) the listening duration measured in log seconds increased by 0.96% and iii) the number of customers who played music increased by 0.92%.

All results for which statistical significance is not specified were significant with p-values below 0.001. Finally, we observed similar to Section 5.2.1 that the ranking bandit was able to capture intra-day trends, where it ranked a summer playlist higher during the day and evening than at night, and sudden customer trends, where it learned within 1 day to rank high a banner featuring a new track by a famous American artist. In both cases, the matrix factorization baseline missed these trends. See Figures 2c and 2d for more details.

6 LESSONS LEARNED

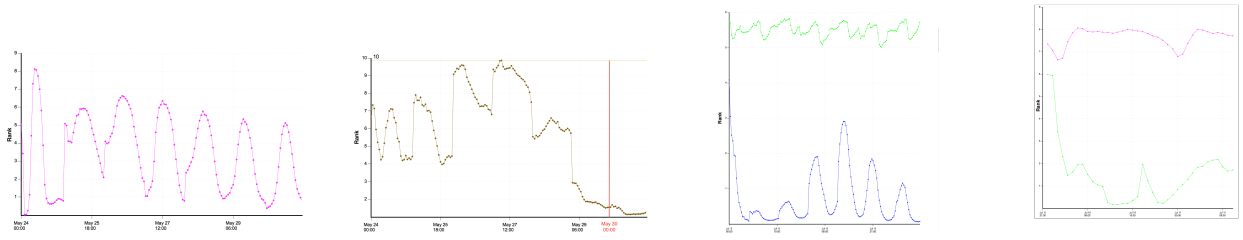
While the experiments turned out successful, there are a few facts which we considered surprising.

6.1 Usage of one-hot encoding

We developed methods which leverage "contextualized actions" allowing us to perform an extensive amount of feature engineering. In this way we can leverage highly non-linear model trained on historical information to produce high-quality features. In the online experiments we reported in this paper, we used the our system to re-rank a very small pool of items (represented by a large image) linked to a piece of musical content. Turns out that the one-hot-encoding representation of the items combined with the context by the mean of the cross product and a non-linear dimensionality reduction technique performed very well. We do not have a scientific explanation of the reasons behind this success, but we conjecture that the visual aspect of the items plays a crucial role which is hard to capture in a small set of visual features. Moreover, the small content pool compared to the number of requests served allows the algorithm to converge quickly also without information about the similarity between the actions. To verify the contribution of the visual aspects to customers' decisions and the best way to encode the images associated to musical items is left as future work.

6.2 Position bias estimation

As reported in the previous section, we tested the Thompson Sampling ranking algorithm online in combination with the automatic position bias estimation leveraging expectation maximization (previously called LinTS-PBMRank(EM)). While we obtained positive results in the online experiment, we observed an unexpected behaviour in the probabilities computed by the EM algorithm which could have been related to numerical stability issues and further investigated the matter. We decided to run a new online experiment where LinTS-PBMRank(EM) was compared with an instance of the same algorithm whose position bias probabilities where manually tuned leveraging historical data. This experiment terminated with a significant victory (about 5% increase in MRR) of the algorithm using manually tuned position bias. Re-applying offline part of the updates to the model, we noticed that even using a consistent number of updates, in the order of 10^5, the posteriors means of the two models were not converging to the same value. Specifically, their cosine similarity was in the interval (0.6, 0.8). This is due to: i) the random initialization of the EM model and ii) the error made by the predictor in estimating the rewards. We decided to change the initialization of the EM model to $\frac{1}{\ell+\epsilon}$ where ϵ is just a small random number (e.g., in (0, 0.1)). The same offline analysis described above



(a) (b) (c) (d)

Figure 2: a) Intra-day trends for audio content of niche genre. b) Intra-day and intra-week trend for an item regarding music for Father's day. There is an evident trend in the days leading to the holiday where the item becomes more popular. c) Intra-day trends for summer playlist in the beginning of July. The ranking bandit is in blue and the baseline recommender in green. The ranking bandit is able to catch the general trend earlier than the baseline recommender and also to follow the intra-day fluctuations. d) Trend for for banner featuring a new track by a well-known American singer. The ranking bandit is in green and the baseline recommender in pink. As it often happens recently released content by popular artists catches the attentions of customers outside the core artist fan base. In this plot it is evident that the ranking bandit catches the trend much earlier than the baseline recommender.

provides significantly different results using this initialization, with an average cosine similarity of the posterior means at 0.93 and negligible variance. We tested a few initialization techniques offline with slightly worse but comparable results and we are waiting to validate our findings in online experiments.

7 CONCLUSIONS AND FUTURE WORK

We provided extensions of two well-known contextual bandit algorithms that show a significant empirical advantage in real-world scenarios. Our online experiments were run on a large scale music streaming service show a significant customer impact measured by a few different metrics. Moreover, the presented algorithms proved themselves easy to maintain in a production environment.

There are a few directions in which we are considering extending these ranking solutions: i) perform additional experiment on the most effective representations to be used for music recommendations in visual clients, ii) scale known techniques [11, 12] for the multi-bandit setting to support a massive number of customers iii) compare our results with the ones obtained by more complex solutions based on complex reinforcement learning algorithms.

REFERENCES

[1] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pages 2312–2320, 2011.

[2] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *ICML '13*, pages 127–135, 2013.

[3] Nicolo Cesa-Bianchi and Gábor Lugosi. Combinatorial bandits. *Journal of Computer and System Sciences*, 78(5):1404–1422, 2012.

[4] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the 14. International Conference on Artificial Intelligence and Statistics*, pages 208–214, 2011.

[5] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. Click models for web search. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 7(3):1–115, 2015.

[6] Richard Combes, Mohammad Sadegh Talebi Mazraeh Shahi, Alexandre Proutiere, et al. Combinatorial bandits revisited. In *Advances in Neural Information Processing Systems*, pages 2116–2124, 2015.

[7] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. An experimental comparison of click position-bias models. In *Proceedings of the 2008 international conference on web search and data mining*, pages 87–94. ACM, 2008.

[8] Paolo Dragone, Rishabh Mehrotra, and Mounia Lalmas. Deriving user- and content-specific rewards for contextual bandits. WWW '19, page 2680–2686, New York, NY, USA, 2019.

[9] Miroslav Dudik, Daniel Hsu, Satyen Kale, Nikos Karampatziakis, John Langford, Lev Reyzin, and Tong Zhang. Efficient optimal learning for contextual bandits. UAI '11, pages 169–178, Arlington, Virginia, United States, 2011. AUAI Press.

[10] Antonino Freno. Practical lessons from developing a large-scale recommender system at Zalando. In *RecSys '17*, pages 251–259, New York, NY, USA, 2017. ACM.

[11] Claudio Gentile, Shuai Li, Purushottam Kar, Alexandros Karatzoglou, Giovanni Zappella, and Evans Etrue. On context-dependent clustering of bandits. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1253–1262. JMLR. org, 2017.

[12] Claudio Gentile, Shuai Li, and Giovanni Zappella. Online clustering of bandits. In *International Conference on Machine Learning*, pages 757–765, 2014.

[13] Thore Graepel, Joaquin Quinonero Candela, Thomas Borchert, and Ralf Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in Microsoft's bing search engine. In *ICML '10*, 2010.

[14] Alois Gruson, Praveen Chandar, Christophe Charbuillet, James McInerney, Samantha Hansen, Damien Tardieu, and Ben Carterette. Offline evaluation to make decisions about playlist recommendation algorithms. WSDM '19, page 420–428, New York, NY, USA, 2019. Association for Computing Machinery.

[15] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In *ACM SIGIR Forum*, volume 51, pages 4–11. ACM, 2017.

[16] Junpei Komiyama, Junya Honda, and Akiko Takeda. Position-based multiple-play bandit problem with unknown position bias. In *Advances in Neural Information Processing Systems*, pages 4998–5008, 2017.

[17] Branislav Kveton, Csaba Szepesvári, Zheng Wen, and Azin Ashkan. Cascading bandits: Learning to rank in the cascade model. In *ICML'15*, pages 767–776, 2015.

[18] Paul Lagrée, Claire Vernade, and Olivier Cappe. Multiple-play bandits in the position-based model. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1597–1605. Curran Associates, Inc., 2016.

[19] Tor Lattimore, Branislav Kveton, Shuai Li, and Csaba Szepesvári. Toprank: A practical algorithm for online stochastic ranking. In *NeurIPS*, 2018.

[20] Shuai Li, Yasin Abbasi-Yadkori, Branislav Kveton, S Muthukrishnan, Vishwa Vinay, and Zheng Wen. Offline evaluation of ranking policies with click models. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1685–1694. ACM, 2018.

[21] Alexander R. Luedtke, Emilie Kaufmann, and Antoine Chambaz. Asymptotically optimal algorithms for budgeted multiple play bandits. Preprint (https://hal.archives-ouvertes.fr/hal-01338733), 2017.

[22] Rishabh Mehrotra, James McInerney, Hugues Bouchard, Mounia Lalmas, and Fernando Diaz. Towards a fair marketplace: Counterfactual evaluation of the trade-off between relevance, fairness & satisfaction in recommendation systems. CIKM '18, page 2243–2251, New York, NY, USA, 2018.

[23] Matthew Richardson, Ewa Dominowska, and Robert Ragno. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web*, pages 521–530. ACM, 2007.

[24] Jack Sherman and Winifred J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):124–127, 1950.

[25] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. Position bias estimation for unbiased learning to rank in personal search. In *WSDM'18*, pages 610–618. ACM, 2018.

[26] Zheng Wen, Branislav Kveton, and Azin Ashkan. Efficient learning in large-scale combinatorial semi-bandits. In *ICML '15*, pages 1113–1122, 2015.

[27] Shi Zong, Hao Ni, Kenny Sung, Nan Rosemary Ke, Zheng Wen, and Branislav Kveton. Cascading bandits for large-scale recommendation problems. UAI '16, pages 835–844, Arlington, Virginia, United States, 2016. AUAI Press.