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

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.
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 \( q_{\ell} \) denotes the examination probability that position \( \ell \) was observed (also known as position bias). The core assumption of PBM is that the observed click Bernoulli variable \( C \) depends on two 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 \( \ell \). Then, we can compute the expected payoff of each action \( a \)

\[
\mathbb{E}(\hat{\theta} \parallel x, a, \ell) = \left( \sum_{\ell=1}^{L} q_{\ell} \hat{a}_{\ell}^{T} \theta \right) / \left( \sum_{\ell=1}^{L} q_{\ell} \right)
\]

where \( \hat{a}_{\ell} = \arg \min_{\theta} \mathbb{E}[\theta \parallel x, a, \ell] \).

In Section 4, we discuss how we derive \( q_{\ell} \).

The Learning Model. We consider a linear bandit setting in which the ranked actions at each round is a list of L actions chosen from a given set \( \{a_1, \ldots, a_K\} \) 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 \( a_{t} = \{a_{1}^{t}, \ldots, a_{K}^{t}\} \) and the list action selected at time \( t \) will be denoted as \( A_{t} = \{A_{1}^{t}, \ldots, A_{L}^{t}\} \), where \( A_{l} \) is a permutation of \( a_{l} \).

The PBM is characterized by examination parameters \( \beta \) and \( \lambda \). The observed action \( A_{l} \) is a click Bernoulli variable \( C \) depends on two 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 \( \ell \). Then, we can compute the expected payoff of each action \( a \) at position \( \ell \) as

\[
\mathbb{E}(\hat{\theta} \parallel x, a, \ell) = \left( \sum_{\ell=1}^{L} q_{\ell} \hat{a}_{\ell}^{T} \theta \right) / \left( \sum_{\ell=1}^{L} q_{\ell} \right)
\]

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 \( \theta \) using all past observations:

\[
\hat{\theta}_{t} = \arg \min_{\theta \in \mathbb{R}^{d}} \mathbb{E}[\theta \parallel x, a, \ell] = \mathbb{E}[\theta \parallel x, a, \ell]
\]

We can now derive a conditionally unbiased estimator of the model parameter \( \theta \) for the ranking case in the PBM as a least square estimate:

\[
\hat{\theta}_{t} = \mathbb{E}[\theta \parallel x, a, \ell] = \left( \sum_{\ell=1}^{L} q_{\ell} \hat{a}_{\ell}^{T} \right) / \left( \sum_{\ell=1}^{L} q_{\ell} \right)
\]

where \( \ell \) is the least square solution of \( \hat{\theta}_{t} = \arg \min_{\theta \in \mathbb{R}^{d}} \mathbb{E}[\theta \parallel x, a, \ell] = \mathbb{E}[\theta \parallel x, a, \ell] \).

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 \( \theta \). The true observation \( Z_{t}^{\ell} \) is replaced by its conditional expectation given the censored position variables \( Y_{t}^{\ell} \sim \mathbb{B}(q_{\ell}) \). We introduce the filtration \( F_{t} \) as the union of history until time \( t - 1 \), and the context at time \( t \), \( F_{t} = (A_{1}, Z_{1}, \ldots, A_{t}) \) such that for all \( t, \ell, E[Z_{t}^{\ell} \mid F_{t}] = E[C_{t}^{\ell} \mid F_{t}] = q_{\ell} \theta_{t} \). We present a fully Bayesian treatment of Linear Thompson Sampling where we assume \( \sigma^{2} \) follows an Inverse-Gamma distribution and \( \theta \) follows a multivariate Gaussian:

\[
\sigma^{2} \sim IG(a_{0}, b_{0}) = p_{0}(\sigma^{2})
\]

\[
\theta \sim N(\theta_{0}, \sigma^{2}V_{0}^{-1}) = p_{0}(\theta)
\]

\[
Z_{t}^{\ell} | A_{t}, \theta, \sigma^{2} \sim N(\theta_{t} \theta_{t}, \sigma^{2})
\]

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

\[
p(\hat{\theta} \mid F_{t}) \propto p_{0}(\sigma^{2})p_{0}(\theta) \prod_{t=1}^{T} \prod_{\ell=1}^{L} p(Z_{t}^{\ell} \mid \theta_{t}, F_{t})
\]

\[
\propto \exp\left\{ -\frac{1}{2\sigma^{2}} \sum_{t=1}^{T} \left( Z_{t}^{\ell} - q_{\ell} \theta_{t} \right) \right\} \left( \left( \frac{1}{\sigma^{2}} \right)^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2\sigma^{2}} \left( \theta - \theta_{0} \right)^{T}V_{-1}(\theta - \theta_{0}) \right\} \right\}
\]

We rearrange the posterior to formalize the posterior mean \( \theta_{t} \) and the variance \( V_{t}^{-1} \) in closed form. First, we rewrite the quadratic
Algorithm 1 LinUCB-PBMRank

**Input:** Position Bias Parameters \((q_t, \ldots, q_L)\), confidence level \(\delta > 0\), regularization \(\lambda\).

**for** \(t = 1, \ldots, T\) **do**

1. Get the contextualized actions \(\mathcal{A}_t\).
2. Compute \(\hat{\theta}_t\) as in Prop. 1 and for all \(a \in \mathcal{A}_t\),
   \[ U_t(a) = a^T \hat{\theta}_t + \sqrt{f_{t,b} a[a]V_{-1}}. \]
3. Build Top-L action list \(A_t \in \arg \max_{a \in \mathcal{A}_t} \sum_t q_t U_t(a)\),
   (ties broken arbitrarily)
4. Update \(V_t \leftarrow V_{t-1} + \sum_t q_t^2 A_t^T A_t^T\)
5. Receive feedback for round \(t\)
6. Update \(b_t \leftarrow b_{t-1} + \sum_t q_t V_t A_t^T\)

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 \(P(C = 1|d) = \rho_t\). Given the click event is always observed, \(\rho_t\) can be estimated using MLE. The likelihood for the parameter \(\rho_t\) can be written as:

\[
\mathcal{L}(\rho_t) = \prod_{c \in S_t} \rho_t^{c_t}(1 - \rho_t)^{1 - c_t}
\]

where \(S_t\) is the set of clicks and \(c_t\) is the value of the click of the \(t^{th}\) occurrence for position \(t\). By taking the log of (2), calculating its derivative and equating it to zero, we get the MLE estimation of the parameter \(\rho_t\). In this case, it is the sample mean of \(c_t\):

\[
\rho_t = \frac{\sum_{c \in S_t} c_t}{|S_t|}
\]

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:

\[
P(C|\beta, x) = \Phi(C : \beta^T x/\beta),
\]

where we assumed that \(C\) is either 1 (click) or 0 (no click) and \(\Phi\) is the cumulative density function of the standard normal distribution:

\[
\Phi(t) = \int_{-\infty}^{t} N(s; 0, 1) ds.
\]

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 \(\beta\) scales the steepness of the inverse link function. The function \(P(C|\beta, x)\) is called likelihood as a function of \(\beta\) and sampling distribution as a function of \(C\); the latter is the...
We keep a Probit Regression (PR) model for each position $\mathcal{C}$ where the former is the weighting that the data, $\mathcal{P}$, then the predictive distribution $\mathcal{P}$ is calculated as:

$$P(\theta|x,a,t) = P(C|x,a,t,\theta)P(\theta)$$

Then, the predictive distribution $P(C|x,a,t)$ 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 $t$ is equal to $P(C = 1|x,a,t) = P(E = 1|t)P(R = 1|x,a)$. Here, our goal is to compute $q_t = P(E = 1|t)$ and we compute it as:

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

where we assume $P(E = 1|t = 1) = 1$. 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 this type 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:

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

(4)

However, explicitly finding the maximum likelihood estimates of the parameters $\theta$ 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 $y_{x,a} = P(R = 1|x,a)$, the position bias $q_{t} = P(E = 1|t)$ where $P(C = 1|x,a,t) = P(E = 1|t)P(R = 1|x,a)$, and a regular click log $\mathcal{L} = \{(c,x,a,t)\}$, the log likelihood of generating this data is:

$$\log P(\mathcal{L}) = \sum_{(c,x,a,t) \in \mathcal{L}} c \log q_{y_{x,a}} + (1-c) \log (1 - q_{y_{x,a}})$$

(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 $q_{t}$ and the relevance parameter $y_{x,a}$. In this paper, we modify the standard EM and take $y_{x,a}$ equal to $\sigma(A_{x}^{t}\hat{\theta}_{t})$ 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 $\mathcal{L}$:

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

(6)

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

$$q_{t}^{(t+1)} = \frac{1}{t} \sum_{t} c_{t}^{(t)} + (1 - c_{t}^{(t)}) \frac{1 - y_{x,a}^{(t)}q_{t}^{(t)}}{1 - q_{t}^{(t)}y_{x,a}^{(t)}}$$

(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 performed 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 would be irresponsible and completely against customers’ interest. Would be irresponsible and completely against customers’ interest.

### 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 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 output 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
The exponential function was chosen after observing the behaviour of 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.

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.

<table>
<thead>
<tr>
<th>ALGORITHMS</th>
<th>Number of actions/positions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>LinUCB</td>
<td>4827.30 ± 5.99</td>
</tr>
<tr>
<td>LinUCB-PBMRank (Real)</td>
<td>48274.81 ± 3.51</td>
</tr>
<tr>
<td>LinUCB-PBMRank (EM)</td>
<td>4626.04 ± 10.63</td>
</tr>
<tr>
<td>LinUCB-PBMRank (PR)</td>
<td>48276.28 ± 3.51</td>
</tr>
<tr>
<td>LinUCB-PBMRank (CTR)</td>
<td>48247.31 ± 5.90</td>
</tr>
<tr>
<td>LinTS</td>
<td>48513.36 ± 12.93</td>
</tr>
<tr>
<td>LinTS-PBMRank (Real)</td>
<td>48559.73 ± 9.27</td>
</tr>
<tr>
<td>LinTS-PBMRank (EM)</td>
<td>48155.47 ± 49.21</td>
</tr>
<tr>
<td>LinTS-PBMRank (PR)</td>
<td>48597.39 ± 9.27</td>
</tr>
<tr>
<td>LinTS-PBMRank (CTR)</td>
<td>48523.96 ± 6.31</td>
</tr>
<tr>
<td>Random Selection</td>
<td>45044.14 ± 11.60</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>ALGORITHMS</th>
<th>Number of actions/positions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>LinUCB</td>
<td>34790.07 ± 5.53</td>
</tr>
<tr>
<td>LinUCB-PBMRank (Real)</td>
<td>34682.12 ± 31.15</td>
</tr>
<tr>
<td>LinUCB-PBMRank (EM)</td>
<td>34568.03 ± 173.82</td>
</tr>
<tr>
<td>LinUCB-PBMRank (PR)</td>
<td>34584.33 ± 24.68</td>
</tr>
<tr>
<td>LinUCB-PBMRank (CTR)</td>
<td>34552.33 ± 124.94</td>
</tr>
<tr>
<td>LinTS</td>
<td>34850.33 ± 130.12</td>
</tr>
<tr>
<td>LinTS-PBMRank (Real)</td>
<td>34882.33 ± 117.58</td>
</tr>
<tr>
<td>LinTS-PBMRank (EM)</td>
<td>34795.66 ± 113.52</td>
</tr>
<tr>
<td>LinTS-PBMRank (PR)</td>
<td>34882.33 ± 35.12</td>
</tr>
<tr>
<td>LinTS-PBMRank (CTR)</td>
<td>34766.01 ± 171.48</td>
</tr>
<tr>
<td>Random Selection</td>
<td>26721.66 ± 35.42</td>
</tr>
</tbody>
</table>

Table 3: Cumulative reward on SINBIN when position bias is...
while the EM method is performing its estimation without any
we can observe two important trends: i) not addressing the position
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
document.

are often served in web-pages containing several elements which
to visualize the recommendations. This is particularly useful in cases where
we do not need to employ techniques for running offline evaluation
with biased datasets (e.g., [20]). In these offline experiments,
we 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
equality 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.

5.1.1 Results. In our experiments, we compared the two algo-
rithms 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 \( \{q_t\}_t=1 \) to 1 in the
algorithms referenced above.

Synthetic Data Results. Tables 1 and 2 report the results of ex-
periments run on synthetic data in order to validate our ideas in a
controlled environment. The dataset used in this section are SIN-
REAL 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 eval-
uation with biased datasets (e.g., [20]). In these offline experiments,
we 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
equality 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) 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, be-
tween 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
target the ranking, we leverage different types of features
representing the customer, content, and general context, such as
temporal information, customer taste profiles and customer affini-
ties towards musical content. These representations are used as
input for the bandit, which is solving the LTR problem. We see

\[ A \text{ carousel consists of a list of banners, where only one banner is displayed at a time and rotated to the next one after a certain time period.} \]
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 10^6, 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 \( \frac{1}{\sqrt{\epsilon}} \) where \( \epsilon \) is just a small random number (e.g., in (0, 0.1)). The same offline analysis described above
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 showing 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 experiments on the most effective representations to be used for music recommendations in visual clients, ii) scale known techniques [11, 12] for the most effective representations to be used for music recommendations themselves easy to maintain in a production environment.

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 showing 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 experiments on the most effective representations to be used for music recommendations in visual clients, ii) scale known techniques [11, 12] for the most effective representations to be used for music recommendations 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 experiments on the most effective representations to be used for music recommendations in visual clients, ii) scale known techniques [11, 12] for the most effective representations to be used for music recommendations themselves easy to maintain in a production environment.

REFERENCES