Deriving User- and Content-specific Rewards for Contextual Bandits

Paolo Dragone*†
University of Trento
TIM (SKIL), Trento
paolo.dragone@unitn.it

Rishabh Mehrotra
Spotify
rishabhm@spotify.com

Mounia Lalmas
Spotify
mounia@acm.org

ABSTRACT

Bandit algorithms have gained increased attention in recommender systems, as they provide effective and scalable recommendations. These algorithms use *reward functions*, usually based on a numeric variable such as click-through rates, as the basis for optimization. On a popular music streaming service, a contextual bandit algorithm is used to decide which content to recommend to users, where the reward function is a binarization of a numeric variable that defines success based on a static threshold of user streaming time: 1 if the user streamed for at least 30 seconds and 0 otherwise. We explore alternative methods to provide a more informed reward function, based on the assumptions that streaming time distribution heavily depends on the type of user and the type of content being streamed. To automatically extract user and content groups from streaming data, we employ "co-clustering", an unsupervised learning technique to simultaneously extract clusters of rows and columns from a co-occurrence matrix. The streaming distributions within the co-clusters are then used to define rewards specific to each co-cluster. Our proposed co-clustered based reward functions lead to improvement of over 25% in expected stream rate, compared to the standard binarized rewards.

1 INTRODUCTION

Given the overwhelming choices faced by users on what to watch, read and listen to online, recommender systems play a pivotal role in helping users navigate the myriad of choices. Most modern recommender systems are powered by interactive machine learning algorithms such as bandits [14, 17, 23], which learn to adapt their recommendations by estimating a model of the user satisfaction metric from the users feedback.

A key component of such bandit based recommender systems is the choice of *reward function.* A proper reward function should correlate well with user satisfaction, to allow the model to learn to serve good recommendations. Most prior work on quantifying satisfaction has relied on leveraging implicit signals derived from user interactions, mostly clicks [12, 19, 21, 25]. Often, click based signals are not informative enough and fail to differentiate between satisfying and dissatisfying experiences of users [11, 28]. Consequently, there has been a pressing need to move beyond clicks and investigate *post-click* behavior of users. These focus on the engagement of users with the served recommendations and have shown promising results in other areas [1, 11, 15].

Considering the specific use case of music streaming, quantifying such a notion of satisfaction from implicit signals involves understanding the diverse needs of users and their expectations of what is a successful streaming session. Such needs often include how users feel, and the expectations that music recommended to them align with their mood or their intent of the moment [8]. To fulfill these, music streaming services provide users with curated playlists, ranging from ”sleep" to ”run". In addition, and to account for the diverse user interests and plethora of musics, genre playlists have been made available to users, ranging from rap, pop, jazz, to niche ones. As a results, millions of playlists are available to users to listen to based on their intent and needs. Given such a heterogeneity in user needs, and the different intents of content (playlists), it becomes important to consider both user and content behavior to formalize the notion of satisfaction, and in turn design the appropriate reward models to capture these.

A major portion of work done on specifying reward functions rely on manually crafted functions, including rewards based on click-through behavior of users and other positive and negative feedback signals [29]. Some efforts have looked at strategies to improve the reward estimation in dynamic recommendation environments [2]. Overall, most bandit models powering recommendation system employ a rather simple threshold based reward function, mostly determined through click-through behavior of the user: *if user clicks on X, a payout of 1 is incurred and 0 otherwise* [17].

In this work, we go beyond such simple threshold-based formulation. We consider the case of a music streaming service, Spotify, powered by a contextual bandit model [20], and aim at defining appropriate reward functions to optimize the bandit model. We revisit how success is quantified, based on how users *consume* playlists and how playlists *are consumed* by users. Our approach moves from a simple user- and content-agnostic, binary reward model to a more sophisticated reward model, which is aware of the distribution of listening behavior. We highlight the need for jointly considering user and content interaction to define rewards, and leverage insights from co-clustering of users and contents together to define rewards. We obtain a substantial improvement of over 25% in expected stream rate with our proposed co-clustering-based calculation of the reward functions.

[*] This work was carried out while Paolo Dragone was an intern at Spotify.
[†] Paolo Dragone is a fellow of TIM-SKIL Trento and is supported by a TIM scholarship.

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313592

2 BACKGROUND AND MOTIVATIONS

Our work aims to improve on a current bandit algorithm used to select which playlists to display to users on the home of Spotify, a popular streaming service [20].

Rewards for playlist recommendation. Playlist recommendation in music streaming services is often cast as a contextual bandit problem. Given a set of playlists $\mathcal{Y}$, a contextual bandit algorithm has to decide which playlists to recommend to a user $u \in \mathcal{U}$ at a certain time iteration $t \in 1, \ldots, \infty$. At each iteration t, the algorithm has access to a "context", i.e. a feature vector $x_{t,y}$ for each playlist $y \in \mathcal{Y}$. For each recommended playlist y_t, the algorithm receives a reward $\hat{r}_t$ based on the metric being optimized and the behavior of the user. In playlist recommendation, the most frequently employed metric is the *stream rate*, i.e. a binary reward indicating whether the user has streamed the playlist y_t or not. When optimizing for stream rate, the reward signal with which the algorithm is trained is a function of the streaming time the user has listened a recommended playlist for, $\hat{r}_t = \hat{r}(s_t)$. This is a binary reward function that assigns a value of 1 to recommendations y_t for which the streaming time s_t exceeds a given threshold of 30 seconds:[1]

$$\hat{r}(s) = \begin{cases} 0 & \text{if } s \leq 30 \text{ seconds} \\ 1 & \text{if } s \geq 30 \text{ seconds} \end{cases} \tag{1}$$

As the users preferences, and thus their behavior, are unknown and non-deterministic, the contextual bandit algorithm has to trade-off *exploration* for gathering information about the user preferences over the playlist domain and *exploitation* for maximizing the expected reward. The expected reward $\mathbb{E}[r_{t,y}]$, in our case the expected stream rate, of a playlist y in context $x_{t,y}$ is unknown and has to be estimated from data collected in the recommendation process. Thus, the expected reward is modeled as a parametrized function of the type:

$$\mathbb{E}[r_{t,y}] = h(x_{t,y}; \theta_t) \tag{2}$$

The above function can then be learned from some hypothesis class $h \in \mathcal{H}$ (e.g. as a linear model or a neural network), and the previously recommended playlists and the resulting users behaviors. New recommendations are selected at each iteration by sampling according to some policy, i.e. a probability distribution that accounts for some degree of exploration versus exploitation. Commonly used policies encompass an exploitation component that selects the best playlists according to the current model $y_t = \operatorname{argmax}_{y \in \mathcal{Y}} \mathbb{E}[r_{t,y}]$, and an exploration component that selects novel recommendations according to some criteria (e.g. uniform random sampling). As this paper focuses on the definition of novel reward functions, we refer to the literature for details on the algorithmic framework [17].

The reward signal defined in Equation 1 is, effectively, a translation of the click-through rate, used in many other domains, to music recommendation. In this paper we explore the use of the streaming time as a metric of its own and not just as a static threshold. Next we discuss the limitations of the standard stream rate metric.

Stream rate. Using the stream rate as above defined as a reward signal, the algorithm learn an expected reward that models the probability of a user to stream a given content y for an undetermined amount of time (more than a fixed threshold):

$$\mathbb{E}[r_{t,y}] = p(\text{user streams } y) \tag{3}$$

This, however, fails to take into account how the user actually engaged with the recommended content *after* a successful recommendation. In a music recommendation system, we can use the streaming time s_{t,y_t} as a proxy for the user engagement: *the longer a user streams a playlist, the higher the user engagement.*

Even when a recommended playlist is streamed for more than 30 seconds, the user behavior, and thus the streaming time, can still vary substantially, and as such may imply different things. If a user clicks on and streams a playlist once, it does not necessarily mean he or she will do it again in the future. Many streams are "explorative", i.e. the user streams a playlist for a short period to evaluate its content. Other unsuccessful recommendations may also be one-time streams or even mistaken clicks [26].

By employing a stream rate reward signal, the algorithm will not learn to distinguish from genuinely successful recommendations, with which the user engages and is likely to engage in the future, and other kinds of uninformative feedback. This is due to the fact that the algorithm gets rewarded the same amount for any recommended playlist that gets streamed, despite eventual large differences in streaming time. Our objective is, instead, to maximize the engagement of the user with respect to the recommended content. To do so, we take into account the distribution of *streaming time* and define a success metric that rewards the algorithm only when it makes recommendations with which the users *truly* engage. This can be achieved by learning as expected reward modeling the probability of a user being engaged with the recommended playlist, assuming the user streams the playlist:

$$\mathbb{E}[r_{t,y}] = p(\text{user engages with } y \mid \text{user streams } y) \tag{4}$$

By learning this expected reward model, the algorithm is able to estimate which content is more likely to keep the user engaged and, hopefully, increase the chances for the user to stream (and engage with) the content in the future. We propose different reward signals that can be used to learn the above expected reward model.

Streaming time. While taking into account the streaming time distribution is a step forward in defining an engagement-aware reward signal for playlist recommendation, the reward is independent on the user and the content that generated it. This implies that the algorithm will learn to judge streams from any kind of users and playlists with the same criteria. This might be suboptimal because different users exhibit different listening behavior and different playlists get listened in vastly different ways.

Indeed, a study comparing streaming time distributions associated with specific groups of playlists and users to the global distribution (for playlists and users, respectively) highlighted these differences. For instance, the average streaming time across all playlists is 23.75 minutes, compared to 43.28 minutes for "sleep" playlists, i.e. playlists people listen to fall asleep. This is almost double the global average, demonstrating that this type of playlists is listened to in a drastically different way. Now focusing on user types, users who like Jazz music tend to listen to recommended

[1] The 30 seconds threshold is somewhat arbitrary, and it is often determined by unrelated factors.

playlists (all playlists, not just Jazz playlists) for longer than other users, with an average of 27.16 minutes, which is significantly higher than the global average of 20.43 minutes.

This data highlights differences between listening habits of users, as well as different attitudes towards certain types of playlists. This suggests that a reward signal based on a single aka "static" threshold is essentially averaging-out these peculiarities that are key to better generalization over a vastly diverse set of users and contents. For this reason, we also propose an approach to take into account user- and content-specific patterns.

User and content patterns. Ideally, we would have a reward function for each pair of user and content, defined on the streaming behaviour of the user with respect to that content. This method, however, would only work in an ideal situation in which we had plenty of data from each user and each playlist. In practice, we have extremely sparse data, in which each user listens to only a handful of the available playlists. Applying such a method on our data would result in poor generalization to similar users and contents. We propose, instead, to group together users and playlists, on the basis of their streaming time distribution, and to define reward functions for each pair of user group and playlist group.

To extract these groups of users and playlists automatically, we need to jointly cluster users on the basis of their streaming behavior and playlists on the basis of how they get streamed. This can be achieved using *co-clustering*, a technique that simultaneously extract clusters from two different variables based on a joint distribution. We then define reward functions specific to each pair of user cluster and playlist cluster, based on the streaming time distribution of the users and playlists of each co-cluster.

3 DISTRIBUTION-AWARE REWARDS

We motivated the need for rewarding the contextual bandit algorithm with a signal based on the engagement of the user with streamed content, as opposed to a standard reward based on stream rate only. Our approach consists in using the streaming time distribution of our data as a basis for defining reward functions for user engagement, with the assumption that the longer a user streams a playlist, the higher the chances of him or her being engaged with the content.

We present three different reward functions aiming to give a sensible notion of user engagement in the playlist recommendation setting.

Mean-based reward. The first reward function consists in a binary reward that determines whether a user is engaged or not. We define a user being engaged with a playlist when he or she streams it for at least as long as global mean $\hat{\mu}$ of streaming times in our dataset. We reward the algorithm with different values based on whether the user is engaged or not. In formula (where $0 \leq \lambda_1 \leq \lambda_2 \leq 1$):

$$\hat{r}_{\hat{\mu}}(s) = \begin{cases} \lambda_1 & \text{if } s \in [0, \hat{\mu}) \\ \lambda_2 & \text{if } s \geq \hat{\mu} \end{cases} \tag{5}$$

Additive reward. A more fine-grained solution is to define "engagement levels", each associated with a specific interval of streaming time. Given the streaming time s the user spent on listening to a recommended playlist, the algorithm will observe the reward amount associated with the proper interval. We define the engagement levels based on how different the streaming time is from the mean $\hat{\mu}$. The variability of the streaming time can be quantified with the standard deviation $\hat{\sigma}$ of the streaming times, which we can use to define engagement levels in the following way:

$$\hat{r}_{\hat{\mu},\hat{\sigma}}(s) = \begin{cases} \lambda_1 & \text{if } s \in [0, \hat{\mu} - \hat{\sigma}) \\ \lambda_2 & \text{if } s \in [\hat{\mu} - \hat{\sigma}, \hat{\mu}) \\ \lambda_3 & \text{if } s \in [\hat{\mu}, \hat{\mu} + \hat{\sigma}) \\ \lambda_4 & \text{if } s \geq \hat{\mu} + \hat{\sigma} \end{cases} \tag{6}$$

Here $0 \leq \lambda_j \leq \lambda_{j+1} \leq 1$, for $1 \leq j \leq 3$, are the reward amounts associated with each level. With the same approach one could define more engagement levels at any given granularity, by adding and subtracting multiples and submultiples of the standard deviation. In our experiments, however, we considered only these four intervals.

Cumulative reward. We can define *infinite* engagement levels as a *continuous* function of the streaming time, based on the underlying distribution. This function needs to be bounded between 0 and 1 and increases with the streaming time. We obtain this "cumulative" function by first fitting a model to the relative frequency distribution, and then using the associated cumulative distribution function to define the reward. As the streaming time distribution in our data follow a clear exponentially decreasing pattern, we fit the following exponential model to our data:

$$f(s; \gamma) = \gamma e^{-\gamma s} \tag{7}$$

The cumulative distribution function associated with the above is:

$$F(s; \gamma) = 1 - e^{-\gamma s} \tag{8}$$

Once fitted, wee obtain the best $\hat{\gamma}$ value to fit our data, which can be used to parametrize the cumulative distribution function. The cumulative reward function is then defined as follows:

$$\hat{r}_{\hat{\gamma}}(s) = \max \{F(s; \hat{\gamma}), \lambda\} \tag{9}$$

4 USER- AND CONTENT-AWARE REWARDS

We defined rewards that take into account the distribution of the streaming time of the users, thereby optimizing the probability of picking recommendations that would result in high streaming times when streamed. Streaming patterns across different groups users and different group of playlists are, however, very different from each other. In this section we introduce a technique that allows us to subdivide our data into smaller sets associated with closely related groups of users and playlists, on the basis of their streaming behavior. These subsets then help us defining reward functions that are specific to the different user and playlist groups.

Co-clustering. We want to subdivide the distribution into smaller sets based on similar streaming patterns across users and playlists. As the streaming time distribution is jointly dependent on both users and clusters, we should take into account these dependency in the clustering process. This can be achieved using *co-clustering*, a technique that simultaneously clusters two (or more) different variables based on a joint underlying distribution [6, 7, 24, 27]. Co-clustering has been successfully employed in combination with several recommendation techniques [4, 9, 16]. Co-clustering has also been used to improve recommendation based contextual bandits algorithms [18]. The difference with our work is that [18] learns a co-clustering from the given reward, e.g. click-through rate, whereas we use co-clustering to define specific metrics to optimize for each user and playlist group.

We employ the information-theoretic co-clustering technique introduced by Dhillon et al. [7]. Given two random variables, in our case one variable $U \in \mathcal{U}$ representing the user and a variable $Y \in \mathcal{Y}$ representing the playlist, and an underlying joint probability distribution $p(U, Y)$, co-clustering will output two sets of clusters $\hat{\mathcal{U}}$ and $\hat{\mathcal{Y}}$, as well as two mappings $C_U : \mathcal{U} \to \hat{\mathcal{U}}$ and $C_Y : \mathcal{Y} \to \hat{\mathcal{Y}}$ from elements onto clusters. The number of k of row clusters and ℓ of column clusters, i.e. elements of $\hat{\mathcal{U}}$ and $\hat{\mathcal{Y}}$ respectively, is fixed a priori. The aforementioned technique computes a co-clustering by minimizing the loss in mutual information [5] between the original variables and the clustered variables. This in turn equates to finding an approximate distribution $q(U, Y)$ minimizing the KL-divergence with the original distribution:

$$I(U; Y) - I(\hat{U}; \hat{Y}) = D\left(p(U, Y) \| q(U, Y)\right) \tag{10}$$

where $q(U = u, Y = y) = p(\hat{u}, \hat{y})p(u|\hat{u})p(y|\hat{y})$, $C_U(u) = \hat{u}$, $C_Y(y) = \hat{y}$.

The algorithm minimizes the objective in a coordinate descent fashion: at each iteration the algorithm re-computes the row and column clusters separately by minimizing the marginal objectives:

$$C_U(u) = \operatorname*{argmin}_{\hat{u}} D(p(Y|u) \| q(Y|\hat{u})) \tag{11}$$

$$C_Y(y) = \operatorname*{argmin}_{\hat{y}} D(p(U|y) \| q(U|\hat{y})) \tag{12}$$

for each row u (users) and column y (playlists). The algorithm recomputes the distributions $q(\hat{U}, \hat{Y})$, $q(U|\hat{U})$ and $q(Y|\hat{Y})$ with the new found clusters. Each iteration diminishes the joint objective $D(p(U, Y) \| q(U, Y))$. The algorithm keeps iterating through the data until the maximum number of iteration is met or the change in the objective between iterations is lower than a fixed threshold.

We chose this co-clustering technique because it scales better then others with the amount of data we were dealing with. In particular, given that our data is very sparse, this techniques scales as $O(nz \cdot \tau \cdot (k+l))$, where nz is the number of non-zero elements and τ is the number of iterations, which was fixed to 20 as recommended by [7]. We considered a maximum number of 15 clusters for both rows and columns, and leave for future work how to determine the optimal number of co-clusters.

Reward functions based on co-clusters. After co-clustering the users and the playlists, we are left with $k \times l$ co-clusters, each associated with a pair $(\hat{u}, \hat{y}) \in [1, k] \times [1, \ell]$ of user cluster and playlist cluster. Each co-cluster is associated with a set of data points for all users belonging to cluster $\hat{u}$ who listened to playlists belonging to cluster $\hat{y}$. These data points define a co-cluster specific streaming time distribution, which is used to define co-cluster specific reward signals. We use the distribution-aware reward functions defined in Section 3, but with respect to each co-cluster distribution. For a given user u and playlist y belonging, respectively, to cluster $\hat{u}$ and $\hat{y}$, we define the co-cluster specific binary engagement reward:

$$\hat{r}_{\hat{\mu}_{\hat{u},\hat{y}}}(s_i) = \begin{cases} \lambda_1 & \text{if } s_i \in [0, \hat{\mu}_{\hat{u},\hat{y}}) \\ \lambda_2 & \text{if } s_i \geq \hat{\mu} \end{cases} \tag{13}$$

where $\hat{\mu}_{\hat{u},\hat{y}}$ is the average streaming time computed only on the data point contained in the co-cluster $(\hat{u}, \hat{y})$. Similarly, we can also define the additive reward:

$$\hat{r}_{\hat{\mu}_{\hat{u},\hat{y}}, \hat{\sigma}_{\hat{u},\hat{y}}}(s_i) = \begin{cases} \lambda_1 & \text{if } s_i \in [0, \hat{\mu}_{\hat{u},\hat{y}} - \hat{\sigma}_{\hat{u},\hat{y}}) \\ \lambda_2 & \text{if } s_i \in [\hat{\mu}_{\hat{u},\hat{y}} - \hat{\sigma}_{\hat{u},\hat{y}}, \hat{\mu}_{\hat{u},\hat{y}}) \\ \lambda_3 & \text{if } s_i \in [\hat{\mu}_{\hat{u},\hat{y}}, \hat{\mu}_{\hat{u},\hat{y}} + \hat{\sigma}_{\hat{u},\hat{y}}) \\ \lambda_4 & \text{if } s_i \geq \hat{\mu}_{\hat{u},\hat{y}} + \hat{\sigma}_{\hat{u},\hat{y}} \end{cases} \tag{14}$$

with $\hat{\sigma}_{\hat{u},\hat{y}}$ being the cluster specific standard deviation. For the cumulative reward, we can fit different exponential models to each co-cluster, thereby defining the co-cluster based reward as:

$$\hat{r}_{\hat{\gamma}_{\hat{u},\hat{y}}}(s_i) = \max\left\{F(s_i; \hat{\gamma}_{\hat{u},\hat{y}}), \lambda\right\} \tag{15}$$

where $\hat{\gamma}_{\hat{u},\hat{y}}$ is the parameter found by fitting the exponential model on the co-cluster data.

The procedure for getting the co-cluster specific rewards from the data is straightforward. First compute a co-clustering over the dataset and group examples (u_i, x_i, y_i, s_i) by their associated co-cluster $(\hat{u}_i, \hat{y}_i)$. For each co-cluster $(\hat{u}, \hat{y})$ compute the parameters $\hat{\mu}_{\hat{u},\hat{y}}$, $\hat{\sigma}_{\hat{u},\hat{y}}$ and $\hat{\gamma}_{\hat{u},\hat{y}}$. Finally assign the co-cluster specific rewards to each data point employing the respective parameters.

5 EXPERIMENTS

Dataset. The dataset consists of a sample of logged feedback data from user interaction with the recommendation system of Spotify, a popular music streaming service. The datasets contains about 9M data points, each corresponding to an "impression". Each impression is associated with a user u_i, a recommended playlist y_i, a contextual feature vector x_i, and the streaming time s_i (equal to 0 if the playlist was not selected). The data included feedback from more than 800K users about over 900K playlists.

The feature vectors used in the contextual bandit algorithm are composed of more than 150K features, including: (i) features of the user, such as age range, gender, location, affinity to genres; (ii) features of the playlist such as its artist, its (micro and macro) genres, diversity of songs, popularity; (iii) affinity between the user and the playlist, taking into account past interactions, such as streams, skips, likes, and saves; and (iv) other contextual information, such as the day of the week and the time of day.

The data was collected over a period of four weeks from a small percentage of the live traffic of the production system. The first three weeks worth of data were used for the training set (≈ 90%) and the last one for the test set (≈ 10%).

Evaluation Setup. The trained model used in the production system is an ensemble of boosted regression trees, trained by XGBoost [3]. We replicate the same training setup as the production system, including the same hyper-parameters of the learning algorithm. We train a model using the *stream rate* (akin to click-through rates) as reward signal (Equation 1) and use it as a baseline. We also use a *random threshold* approach as a second baseline, which is a binary reward function akin to that in Equation 1, but whose threshold was set to a randomly chosen value in the range [0, 10] minutes, instead of being fixed to 30 seconds.

We compare the two baselines against models trained with the proposed reward functions on the *global* distribution, i.e. using

Reward for global distribution	Prec@1	Prec@5	Rec@1	Rec@5	F_1@1	F_1@5
Mean-based reward	+19.30 %	+3.27 %	+22.68 %	+2.40 %	+20.94 %	+3.13 %
Additive reward	+15.33 %	-0.62 %	+15.81 %	-0.53 %	+15.57 %	-0.60 %
Cumulative reward	+6.91 %	-0.38 %	+8.92 %	-1.04 %	+7.89 %	-0.48 %

Table 1: Precision, recall and F_1 results (percentage over the baseline) for the different rewards over the global distribution.

the same reward function (akin to post-click). We then compare the models trained on reward defined with the use of *co-clustering*, i.e. using the reward specific for its co-cluster (akin to per co-cluster post-click). The hyper-parameters of the reward functions were determined empirically. In addition, as there are no principled way to determine a proper number of clusters, a typical situation when employing unsupervised clustering techniques, the number of row (user) and column (playlist) clusters were also determined empirically. In Tables 1 and 2, we report standard precision, recall and F_1 measures, as improvement percentage over the first baseline (stream rate based reward), for global- and co-clustering-based rewards, respectively.

Evaluating a trained offline model on logged data from an online recommender systems is know to lead to biases in the estimation of the user satisfaction metric. This is because user feedback is collected only on the recommended items selected by the given collection policy, and no feedback is observed on every item that was not recommended. To avoid this problem and perform an *unbiased* evaluation of the models performance, we used a counterfactual evaluation methodology [22], which constitutes a reliable and much less expensive alternative to online A/B testing [10].

We follow the protocol laid out by [17]. We first collect testing data using a random shuffle policy, which assigns a uniform probability score to all recommendations presented to users. Upon evaluation of a new policy, we "re-run" the history of recommendations and keep only the recommendations that happen to be selected again by the policy being evaluated. We then compute the average evaluation metric (expected stream rate) on the retained data points. The evaluation policy used was a multinomial distribution defined on the learned expected reward function. Results comparing various reward strategies are shown (the difference in expected stream rate) in Figures 1 and 2.

6 RESULTS

Comparison of distribution aware rewards. Table 1 presents the precision and recall results comparing distribution aware reward functions with the static binary reward model. Overall, we observe that distribution aware models perform better than considering a fixed threshold. The mean based reward performs better than additive and cumulative reward, with over 19% gain in precision at the top and 22% gain in recall at the top. These results highlight that considering the distribution of user engagement is informative even without considering any user or content specific insights. Furthermore, upon randomly setting the threshold (see Figure 1), we observe an improvement in performance (expected stream rate), which further demonstrates the incompetence of the fixed threshold baseline strategy based on click-through rates.

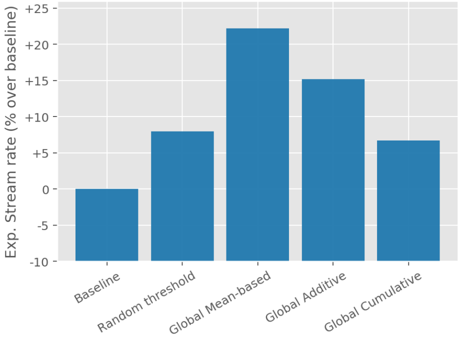

Figure 1: Expected stream rate improvement (in percentage over normalized baseline) of the random threshold method and all reward functions over the global distribution.

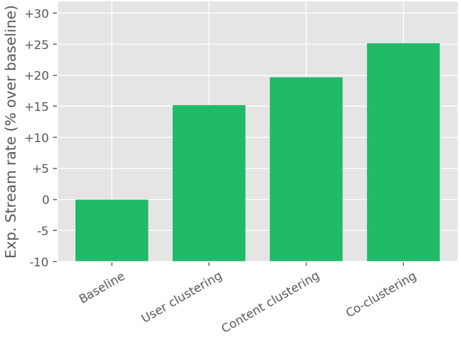

Figure 2: Expected stream rate improvement (in percentage over normalized baseline) of all mean-based reward functions over the distributions obtained by clustering users, clustering playlists and co-clustering.

Using co-clustering. Table 2 presents precision and recall results for the co-clustering method for all three distribution aware rewards. The co-clustering based reward model outperforms all other approaches, i.e. binary reward and global distribution aware reward. We observe a substantial improvement of over 25% in the unbiased expected stream rate metric (see Figure 2). Similar to previously, the mean based function performs better than additive and cumulative even in the case of co-clustering of users and content.

While co-clustering of users and content allows us to jointly capture insights from both when defining the reward function, it may be hard to investigate the potential benefit offered by each. To disentangle the effect of user clusters from content clusters, we consider a special case of clustering, wherein we set the number of content clusters to 1, to give us a user-cluster, and vice-versa for content cluster. As shown in Figure 2, we observe that both

Reward for co-cluster distribution	Row clusters	Column clusters	Prec@1	Prec@5	Rec@1	Rec@5	F_1@1	F_1@5
Mean-based reward	6	13	+24.74 %	+6.11 %	+26.40 %	+5.62 %	+25.56 %	+6.04 %
Additive reward	2	10	+13.34 %	+0.00 %	+12.83 %	+0.31 %	+13.09 %	+0.05 %
Cumulative reward	8	8	+13.34 %	+2.76 %	+14.76 %	+2.58 %	+14.04 %	+2.74 %

Table 2: Precision, recall and F_1 results (percentage over the baseline) for the different rewards over the co-cluster distributions. We experimented with several row x column numbers and we report those leading to the best performance.

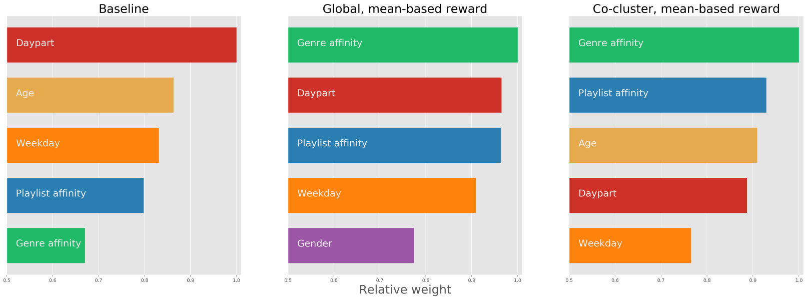

Figure 3: Relative weights of the model's features (by feature type) across the different trained models.

the user- and content- aware reward models perform better than the static binary reward formulation, but both underperform the co-clustering based reward functions.

Comparison across metrics. The counterfactual method results indicate a consistent trend in performance across the unbiased counterfactual and offline evaluation metric, as we observe improvements in co-clustering based reward across all metrics. Furthermore, it is important to note that our estimate of expected stream rate metric is de-biased and corrected from position based bias, and as a result, provides a more reliable indicator of the benefit offered by the joint user- and content- aware reward model.

Quantifying the impact of contributing factors. We analyze how the trained models are different in terms of the features they give importance to. Figure 3 highlights the top 5 feature groups for the baseline binary reward model, the global distribution aware (mean) reward and the co-clustering reward model. As is expected, for the most important feature, the models give different weights to the different feature groups, with the baseline binary model method emphasizing the part of the day features more over others, whereas the distribution aware and user/content aware method focus on genre affinity feature group most. Indeed, the binary reward model is agnostic to user and content types, while the reward function in the co-clustering model is taking into account how users interact with content.

Looking beyond the most important feature group, among the top 5 most important features, user and content co-clustering values features that quantify user- and content- interactions: genre affinity and 7 day card affinity, whereas the baseline binary reward model values part of day and age group of users. Long term interactions like the 30 day affinity of users with content is weighted more in co-clustering than in others, which highlight the fact that training via co-clustering based reward signal allows the model to capture longer term interest and usage patterns.

Finally, we observe that as the reward signal becomes more sophisticated (binary ⟶ global mean ⟶ co-clustering), the focus moves from user-agnostic features (e.g. part of day), to user-only and content-only (e.g. user age and gender), to more sophisticated interaction based features (e.g. 30 day card affinity, genre affinity). These results highlight the importance of carefully defining reward signals that acknowledge the interaction between users and contents in our context (listening to music) by learning to weight the features appropriately.

7 CONCLUSIONS

We motivated this work by posting the hypothesis that distribution aware as well as user- and content-aware reward functions –capturing the "post-click" experience– are better than binary rewards –capturing only clicks. Extensive large-scale experiments on real world user data demonstrated the benefit of *personalizing* the definition of reward function according to how users listen to playlists **and** to how playlists are listened to.

Reward definitions lie at the forefront of systems that adopt reinforcement learning based methods to serve recommendations to users. As can be seen from our results, the reward function governs how the model is trained and has a major impact on the recommendations served. We contend that these results highlight the benefit of considering distribution and user/content aware functions to quantify the rewards, rather than creating hand-crafted reward functions. Setting fixed thresholds is very common. Indeed, a major portion of industrial search systems consider 30 seconds as the fixed threshold on dwell time to gauge user satisfaction with a clicked document [13].

Our findings reinforce the insight that when we optimize for a metric of user satisfaction, it is important to go beyond clicks. By focusing on the post-click experience, we can capture the "*intent*" of the content, as shown with "*genre*"-affinity being the most significant factor in impacting the bandit algorithm. We are also able to capture that users are different regarding how they consume content, as we see that "*playlist affinity*" is often the second most significant factor. Using clicks alone cannot surface these important components when measuring satisfaction.

REFERENCES

[1] Nicola Barbieri, Fabrizio Silvestri, and Mounia Lalmas. 2016. Improving post-click user engagement on native ads via survival analysis. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 761–770.

[2] Shi-Yong Chen, Yang Yu, Qing Da, Jun Tan, Hai-Kuan Huang, and Hai-Hong Tang. 2018. Stabilizing Reinforcement Learning in Dynamic Environment with Application to Online Recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &#38; Data Mining (KDD '18)*. ACM, New York, NY, USA, 1187–1196. https://doi.org/10.1145/3219819.3220122

[3] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 785–794.

[4] Sungwoon Choi, Heonseok Ha, Uiwon Hwang, Chanju Kim, Jung-Woo Ha, and Sungroh Yoon. 2018. Reinforcement Learning based Recommender System using Biclustering Technique. In *WSDM Workshop on Multi-dimensional Information Fusion for User Modeling and Personalization*.

[5] Thomas M Cover and Joy A Thomas. 2012. *Elements of information theory*. John Wiley & Sons.

[6] Inderjit S Dhillon. 2001. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 269–274.

[7] Inderjit S Dhillon, Subramanyam Mallela, and Dharmendra S Modha. 2003. Information-theoretic co-clustering. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 89–98.

[8] Jean Garcia-Gathright, Brian St Thomas, Christine Hosey, Zahra Nazari, and Fernando Diaz. 2018. Understanding and Evaluating User Satisfaction with Music Discovery. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 55–64.

[9] Thomas George and Srujana Merugu. 2005. A scalable collaborative filtering framework based on co-clustering. In *Data Mining, Fifth IEEE international conference on*. IEEE, 4–pp.

[10] Alexandre Gilotte, Clément Calauzènes, Thomas Nedelec, Alexandre Abraham, and Simon Dollé. 2018. Offline A/B testing for Recommender Systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 198–206.

[11] Qi Guo and Eugene Agichtein. 2012. Beyond dwell time: estimating document relevance from cursor movements and other post-click searcher behavior. In *Proceedings of the 21st international conference on World Wide Web*. ACM, 569–578.

[12] Ahmed Hassan, Xiaolin Shi, Nick Craswell, and Bill Ramsey. [n. d.]. Beyond clicks: query reformulation as a predictor of search satisfaction. In *CIKM 2013*.

[13] Youngho Kim, Ahmed Hassan, Ryen W White, and Imed Zitouni. 2014. Comparing client and server dwell time estimates for click-level satisfaction prediction. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 895–898.

[14] Branislav Kveton, Zheng Wen, Azin Ashkan, Hoda Eydgahi, and Brian Eriksson. 2914. Matroid Bandits: Fast Combinatorial Optimization with Learning. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence (UAI'14)*. AUAI Press, Arlington, Virginia, United States, 420–429. http://dl.acm.org/citation.cfm?id=3020751.3020795

[15] Dmitry Lagun and Mounia Lalmas. 2016. Understanding user attention and engagement in online news reading. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 113–122.

[16] Kenneth Wai-Ting Leung, Dik Lun Lee, and Wang-Chien Lee. 2011. CLR: a collaborative location recommendation framework based on co-clustering. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 305–314.

[17] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. 2011. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 297–306.

[18] Shuai Li, Alexandros Karatzoglou, and Claudio Gentile. 2016. Collaborative filtering bandits. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 539–548.

[19] Yiqun Liu, Ye Chen, Jinhui Tang, Jiashen Sun, Min Zhang, Shaoping Ma, and Xuan Zhu. 2015. Different users, different opinions: Predicting search satisfaction with mouse movement information. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 493–502.

[20] James McInerney, Benjamin Lacker, Samantha Hansen, Karl Higley, Hugues Bouchard, Alois Gruson, and Rishabh Mehrotra. 2018. Explore, exploit, and explain: personalizing explainable recommendations with bandits. In *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*. 31–39.

[21] Rishabh Mehrotra, Ahmed Hassan Awadallah, Milad Shokouhi, Emine Yilmaz, Imed Zitouni, Ahmed El Kholy, and Madian Khabsa. 2017. Deep Sequential Models for Task Satisfaction Prediction. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 737–746.

[22] Thomas Nedelec, Nicolas Le Roux, and Vianney Perchet. 2017. A comparative study of counterfactual estimators. *arXiv preprint arXiv:1704.00773* (2017).

[23] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. 2008. Learning Diverse Rankings with Multi-armed Bandits. In *Proceedings of the 25th International Conference on Machine Learning (ICML '08)*. ACM, New York, NY, USA, 784–791. https://doi.org/10.1145/1390156.1390255

[24] Hanhuai Shan and Arindam Banerjee. 2008. Bayesian co-clustering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 530–539.

[25] Ning Su, Jiyin He, Yiqun Liu, Min Zhang, and Shaoping Ma. 2018. User Intent, Behaviour, and Perceived Satisfaction in Product Search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM '18)*. ACM, New York, NY, USA, 547–555. https://doi.org/10.1145/3159652.3159714

[26] Gabriele Tolomei, Mounia Lalmas, Ayman Farahat, and Andrew Haines. 2018. You must have clicked on this ad by mistake! Data-driven identification of accidental clicks on mobile ads with applications to advertiser cost discounting and click-through rate prediction. *International Journal of Data Science and Analytics* (2018).

[27] Pu Wang, Carlotta Domeniconi, and Kathryn Blackmond Laskey. 2009. Latent dirichlet bayesian co-clustering. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 522–537.

[28] Xing Yi, Liangjie Hong, Erheng Zhong, Nanthan Nan Liu, and Suju Rajan. 2014. Beyond clicks: dwell time for personalization. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 113–120.

[29] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with Negative Feedback via Pairwise Deep Reinforcement Learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &#38; Data Mining (KDD '18)*. ACM, New York, NY, USA, 1040–1048. https://doi.org/10.1145/3219819.3219886