Explore, Exploit, and Explain: Personalizing Explainable Recommendations with Bandits

James McInerney, Benjamin Lacker, Samantha Hansen, Karl Higley, Hugues Bouchard, Alois Gruson, Rishabh Mehrotra
Spotify
{jamesm,benl,slhansen,karlhigley,hb,agruson,rishabhm}@spotify.com

ABSTRACT

The multi-armed bandit is an important framework for balancing exploration with exploitation in recommendation. Exploitation recommends content (e.g., products, movies, music playlists) with the highest predicted user engagement and has traditionally been the focus of recommender systems. Exploration recommends content with uncertain predicted user engagement for the purpose of gathering more information. The importance of exploration has been recognized in recent years, particularly in settings with new users, new items, non-stationary preferences and attributes. In parallel, explaining recommendations ("recsplanations") is crucial if users are to understand their recommendations. Existing work has looked at bandits and explanations independently. We provide the first method that combines both in a principled manner. In particular, our method is able to jointly (1) learn which explanations each user responds to; (2) learn the best content to recommend for each user; and (3) balance exploration with exploitation to deal with uncertainty. Experiments with historical log data and tests with live production traffic in a large-scale music recommendation service show a significant improvement in user engagement.

CCS CONCEPTS

• **Information systems → Recommender systems**; **Collaborative filtering**; • **Computing methodologies → Causal reasoning and diagnostics**; **Reinforcement learning**;

ACM Reference Format:
James McInerney, Benjamin Lacker, Samantha Hansen, Karl Higley, Hugues Bouchard, Alois Gruson, Rishabh Mehrotra. 2018. Explore, Exploit, and Explain: Personalizing Explainable Recommendations with Bandits. In *Twelfth ACM Conference on Recommender Systems (RecSys '18), October 2–7, 2018, Vancouver, BC, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3240323.3240354

1 INTRODUCTION

Recommender systems are central to the effectiveness of many online content and e-commerce platforms. Users are overwhelmed by the choice of what to watch, buy, read, and listen to online and recommendations are a popular way to help them navigate this choice. But recommendations without context lack motivation for a user to pay attention to them. Adding an associated explanation for a recommendation (abbreviated as *recsplanation* [16]), is known to increase user satisfaction and the persuasiveness of recommendations [6, 13, 24]

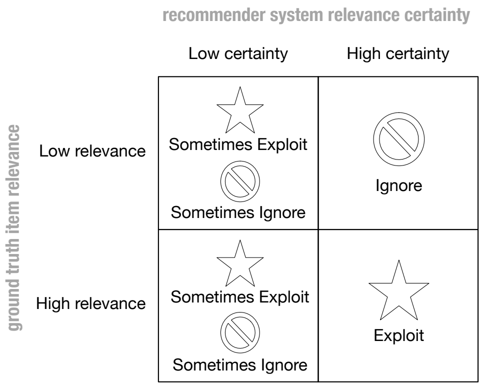
Explanations in recommender systems have the goals of both providing information about recommended items and encouraging better outcomes such as higher user engagement resulting from more confidence and transparency in the recommendations [6]. In this paper, we argue that users respond to explanations differently and that, as a result, there is a need to jointly optimize both item selection and explanation selection for such systems. For example, more social users may respond better to explanations that reference activity in their social network than more private users. In addition, the same users may respond to explanations differently according to their context and intent, e.g., people on the move may require more dependable recommendations from their recent consumption history due to lack of time. We formalize the problem of jointly optimizing item and explanation recommendation and address the technical challenges this problem presents.

In more detail, introducing explanations to recommendation greatly expands the search space of actions the recommender can select from and compounds the problem of data sparsity. Recommendation methods that aim to maximize engagement without regard for model certainty or exploration can unnecessarily ignore highly relevant items. Figure 1a shows the various outcomes when a recommender can only exploit or ignore an item. When there is high certainty about the item relevance, resulting from an abundance of data about the user-item match, the recommender behaves optimally. However, in the face of uncertainty from only a small amount of data, the recommender will sometimes suboptimally ignore relevant items (the lower left quadrant in the grid). The fact that the recommender itself is active in deciding its own training data perpetuates this problem. The problem of ignoring items or whole categories of items leads to filter bubble pathologies [3].

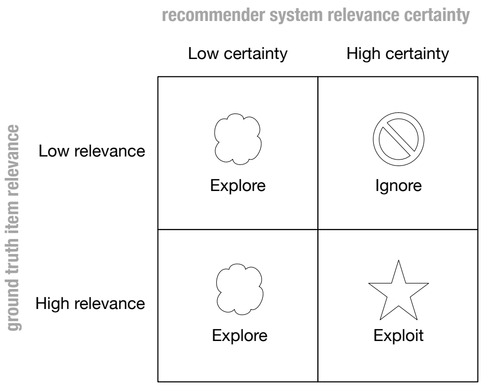
Bandit approaches introduce the notion of exploration to reduce the uncertainty about the relevance of an item [22]. By reserving judgement about item relevance until enough data has been collected, the bandit is able to discover more relevant items (the lower left quadrant of Figure 1b).

The aforementioned issues around item uncertainty also apply to explanation uncertainty. How to incorporate exploration when jointly optimizing for items and explanations is another gap in existing work. Naïvely treating each (*item*, *explanation*) pair as a distinct action would multiplicatively scale the number of actions that need

ACM ISBN 978-1-4503-5901-6/18/10…$15.00
https://doi.org/10.1145/3240323.3240354



(a) Exploitative methods have two modes: exploit or ignore an action.



(b) Bandit methods have three modes: exploit, ignore, or explore an action.

Figure 1: Recommendation methods that aim to maximize engagement without regard for model certainty or exploration sometimes unnecessarily ignore highly relevant items.

to be explored. In addition, in some cases the recommender is required to provide multiple items within the same explanation (e.g., in the shelf layout on a website).

Our approach, BAndits for Recsplanations as Treatments (*Bart*), addresses these gaps in the following way. Bart learns and predicts satisfaction (e.g., click-through rate, consumption probability) for any combination of item, explanation, and context and, through careful logging and contextual bandit retraining, can learn from its mistakes in an online setting.

In summary, our contributions are the following:

- We identify and provide a formalization for the problem of jointly personalizing explainable recommendations.
- We present Bart, a contextual bandits-based framework that addresses the problem of recommending with explanations under uncertainty of user satisfaction.
- Implementing Bart in a scalable production environment presented challenges that we discuss and address.
- Through offline experiments on randomized historical data and online experiments with users in the homepage of a large-scale music recommender system, we empirically evaluate Bart and find that it significantly outperforms the best static ordering of explanations.

This paper is organized as follows. We consider work related to recommendations with explanations and bandits in Section 2. In Section 3, we formalize the problem and explain how Bart addresses it. We evaluate Bart in Section 4 using data offline and in a live production test. Finally, conclusions and future work are discussed in Section 5.

2 RELATED WORK

Our work on personalizing explainable recommendations with bandits builds upon a significant amount of related work both in explaining recommendations and bandits for recommender systems.

Explanations in Recommendations. Explanations help the user better understand and interpret the rationale of the recommender system, thereby making it more trustworthy and engaging. Kouki *et al.* [13] propose a hybrid recommender system built on a probabilistic programming language and show that explanations improve the user experience of recommender systems. Friedrich *et al.* [6] propose a taxonomy of explanation approaches, taking into account the style (e.g., collaborative, knowledge, utility or social explanation style) and paradigm (e.g. content-based, knowledge or collaborative based) and type of preference model. Further, various visualization techniques for explaining recommendations have been proposed, including interfaces with concentric circles [11] and pathways between columns [2].

Bandits for Recommendations. Radlinski *et al.* propose a top-K learning to rank approach using bandits [18]. Li *et al.* consider personalized news recommendation as a contextual bandit problem and propose using LinUCB to evaluate bandit algorithms from logged events [17]. Q-learning and MDPs have been used to recommend items of interest to users, with previous work focusing on recommending news [20], travel information [21], and web pages [9]. Wang *et al.* formulate interactive, personalized recommendation as a bandit task [25]. More recently, the combinatorial problem in bandits was addressed by Kveton *et al.* [14, 15] and Swaminathan *et al.* [23] as *slate* recommendation.

3 METHOD

We next discuss the explainable recommendation problem in more detail in Section 3.1 and present Bart in Section 3.2.

3.1 Problem Formulation

The problem we address in this paper is how to jointly personalize recommendations of items with their associated explanation. Personalization is based on contextual features describing what the system knows about the user, their context, and the item.

We formalize the problem in the following way. Let $\mathcal{J}$ be the set of items, $\mathcal{U}$ be the set of users, and $\mathcal{E}$ the set of explanations. Introduce the validity function $f : \mathcal{E} \times \mathcal{U} \rightarrow 2^{J}$ that maps any pair

Table 1: Examples of parameterized explanation rules

Parameterized Explanation	Rule for Generating Items	Example P's
Movies your friends love	Movies that are popular with P's friends	$\{ user_{123}, user_{921} \}$
More like P	Playlists by artists similar to P	{ David Bowie, Jay-Z, Lady Gaga }
Jump back in	Book genres that a user consumed heavily 6 months ago	$\emptyset$

Examples of simple parameterized rules for specifying the validity function f. Some of these rules assume side information that is commonly accessible, such as a way of finding similar artists through a graph or embedding representation.

Table 2: Summary of symbols

Symbol	Meaning
$j \in \mathcal{J}$	Item to be recommended
$e \in \mathcal{E}$	Explanation for an item
$u \in \mathcal{U}$	User
$x \in \mathcal{X}$	Context, such as user and item attributes
$f(e, u)$	Validity function for explanation e and user u
θ	Coefficients in logistic regression
v	Set of embeddings in factorization machine
X	Context as a random variable
A	Action (i.e. recommendation and explanation) as a random variable
R	Reward as a random variable (e.g. binary consumption indicator)
x_n	Observed context for data point n
a_n	Action performed for data point n
r_n	Observed reward for data point n
N	Total number of data points
D	Dimensionality of context
π	Policy distribution over actions
π_c	Collection policy used to collect the training data, also known as the propensity score
$r^{(t)}$	Reward model with tth order interactions

of elements $e \in \mathcal{E}$ and $u \in \mathcal{U}$ to a subset of $\mathcal{J}$. In other words, f describes the set of valid items $V_{e,u} \subseteq \mathcal{J}$ for each explanation and user.

Why not have all items associated with all explanations, $V_{e,u} \equiv \mathcal{J}, \forall e \in \mathcal{E}, \forall u \in \mathcal{U}$? While this is permissible in the specification of f, it is not a likely outcome because only some items are compatible with each explanation. For example, "because you recently read historical fiction" is only a valid explanation for recommending more historical fiction or related content. We limit our scope to assuming that f has been pre-specified by a designer. Note that this still admits a large number of possible explanations, through simple rules that are parameterized. Table 1 gives several examples of simple rules that can scalably cover the entire item set $\mathcal{J}$ and explanation set $\mathcal{E}$ through parameterization. Finally, note that since we have not restricted f to be injective, there may be more than one explanation for the same item.

Building on these definitions, the problem we address is that of building *a machine learning approach that sorts both explanations and items in a personal, dynamic, responsive, and relevant fashion.* Formally, we seek a reward function $r : \mathcal{J} \times \mathcal{E} \times \mathcal{X} \to \mathbb{R}$ that accurately predicts the user engagement for item $j \in \mathcal{J}$, explanation $e \in \mathcal{E}$, given context $x \in \mathcal{X}$. The goal of training is to optimize the learning objective (which is related to the accuracy of the predicted reward) with respect to the free parameters of r.

In addition, given a reward function, we seek a method for selecting an item-explanation pair (j, e) to present to the user. We refer to this presentation as the *action*, in line with bandit terminology. Choosing the optimal action $(j^*, e^*) = \arg_{j,e} \max r(j, e, x)$ in any particular context x is the exploitative action, but as explained in Section 1, exploiting at every time step is not optimal due to the fact that r is only an estimator of the true reward function. We therefore seek a *policy* $\pi(\cdot|x)$ that is a distribution over actions from which the bandit samples each action in response to the context.

Note that this definition of action and policy makes two independence assumptions. Specifically, it says that the choice of action depends only on the context. Furthermore, it says that the reward is independent of rewards and actions at other time steps given the current action and current context (see Section 3.2). The former assumption is weaker than one would initially suppose, due to the fact that we are free to define the context any way we like. For example, the context can include summary statistics about recent actions or outcomes. The latter is the stronger assumption and naïvely assumes that combinatorial actions such as slate recommendation [23] result in rewards that are a sum of rewards for each action (see Section 5 for a discussion about future work).

With this definition of the problem to be solved we next present Bart, a contextual bandits approach to recommending with explanations. A summary of symbols used in this paper is given in Table 2.

3.2 Bart

The purpose of Bart is to learn how items and explanations interact within any given context to predict user satisfaction. It does this in a reinforcement learning setting where it must decide which actions to take next to gather feedback. Our overall strategy with Bart is to build an effective model of satisfaction from sparse inputs and to use the contextual bandit framework to manage exploration and training.

The standard multi-armed bandit (MAB) maintains a model of the reward R which is conditioned on the choice of action A [22]. The choice of A depends only on the history of actions and rewards up to the current time step $\mathcal{D}_n = \{(a_1, r_1), \ldots, (a_n, r_n)\}$.

The goal of MAB is to choose actions that maximize the total sum of rewards $\sum_{n=1}^{N} r_n$. This problem is reduced to that of:

(1) maintaining an accurate belief about the reward of each action; and,
(2) creating a distribution over actions (i.e., a policy) that effectively balances exploitation (choosing high reward actions) and exploration (finding high reward actions).

There exist several frameworks for doing (2) to varying degrees of sophistication[1] such as epsilon-greedy, upper confidence bound, and Thompson sampling.

The assumptions of the reward model in a standard MAB are summarized in Figure 2a and lead to simple procedures for maintaining a reward belief for each action, e.g., by calculating the posterior distribution $p(R|\mathcal{D}_n)$ given a history of action-reward data $\mathcal{D}_n$ or maintaining a mean and variance estimate for each action.

A criticism of the standard MAB is that it ignores contextual information, denoted by X, that could affect the reward. For example, in recommendation the context may include user features, item features, time of day, platform etc.. This motivates collecting and observing X before each action and making the choice of which action to take dependent on X. This is known as a contextual bandit [1], illustrated in Figure 2b.

There are four important elements to the contextual bandit: the context, reward model, training procedure, and exploration-exploitation policy. We describe each below.

3.2.1 *Context & Reward Model.*

The reward model represents the predicted user engagement resulting from an explained recommendation within a given context. As highlighted in Section 3.1, the cardinality of this input space is $J \times E \times \mathcal{X}$ which is far larger than can be explored exhaustively. We therefore must resort to making assumptions about the structure of the reward function r.

One of the simplest assumptions imposes linearity on the reward model,

$$r(j, e, x) = \sigma(\theta_{\text{global}} + \theta_j^\top 1_j + \theta_e^\top 1_e + \theta_x^\top x), \quad (1)$$

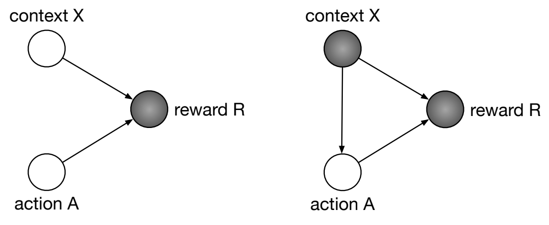
where 1_i represents a one-hot vector of zeros with a single 1 at index i. Eq. 1 combined with a cross-entropy loss function is a logistic regression model of a binary reward, such as whether or not an item was consumed. Binary or consumption count outcomes are common in implicit recommendation [8, 12]. We focus on binary outcomes in this paper for simplicity but note that it can often be replaced with more informative measures of user satisfaction by adapting the last step in the objective function accordingly (e.g., replacing sigmoid σ and cross-entropy loss with identity and a root mean squared error loss function for continuous satisfaction measures).

For convenience, we stack all the inputs and denote them by a single augmented input vector $x' = [1_j^\top, 1_e^\top, x^\top]^\top$ in the same way that we stack all the corresponding parameters $\theta = [\theta_j^\top, \theta_e^\top, \theta_x^\top]^\top$. The logistic regression model is then defined as

$$r^{(1)}(j, e, x) = \sigma(\theta_{\text{global}} + \theta^\top x'), \quad (2)$$

where the aggregated context is now x' and contains information about the item, explanation, and context.

[1]N.B. an alternative formulation of MAB skips problem (1) and attempts to learn an optimal policy directly.



(a) Bandit **(b) Contextual bandit**

Figure 2: Graphical model notation of the multi-armed bandit and the contextual multi-armed bandit. Nodes indicate random variables, arrows direct conditional dependency, and shaded nodes are observed random variables.

Logistic regression has the advantages of interpretability, identifiability, and efficiency. Its major disadvantage in a recommendation setting is the fact that the choice of item and explanation (j, e) that maximizes $r(j, e, x)$ is the same regardless of user or user attributes due to the fact that their contribution to the overall predicted engagement is restricted to be linear. It is therefore a type of *controlled popularity* approach that provides a single set of recommendations for all users in all user contexts where the item and explanation popularity (θ_j, θ_e) are estimated by controlling for other effects.

To get more personalized recommendations, we can introduce weighted sums of higher order interactions between elements in the aggregated context vector x'. When the weights are inner products of latent embeddings, the resulting model family is known as the factorization machine [19],

$$r^{(2)}(j, e, x) = \sigma(\theta_{\text{global}} + \theta^\top x' + \sum_{a=1}^{D} \sum_{b>a}^{D} v_a{}^\top v_b x'_a x'_b) \quad (3)$$

$$r^{(3)}(j, e, x) = \sigma(\theta_{\text{global}} + \theta^\top x' + \sum_{a=1}^{D} \sum_{b>a}^{D} v_a{}^\top v_b x'_a x'_b + \sum_{a=1}^{D} \sum_{b>a}^{D} \sum_{c>b}^{D} < v_a, v_b, v_c > x'_a x'_b x'_c), \quad (4)$$

where D is the dimensionality of x, we introduce latent embeddings v, and $< x, y, z >$ represents the sum of element-wise products of vectors x, y, z. In general, we refer to $r^{(t)}$ as a t^{th}-order factorization machine.

3.2.2 *Off-Policy Training.*

We use counterfactual risk minimization (CRM) to train the contextual bandit [10]. CRM uses importance sample reweighting to account for the fact that the data come from the production policy π_c and not a uniform random experiment. After the reweighting, the optimal parameters $(\hat{\theta}, \hat{v})$ can be found by maximizing likelihood,

$$\hat{\theta}, \hat{v} = \arg_{\theta, v} \max \mathbb{E}_{A \sim \text{Uniform}(\cdot)} [\mathbb{E}_{X,R}[\log p_{\theta,v}(R|A, X)]] \quad (5)$$

$$\approx \arg_{\theta, v} \max \frac{1}{N} \sum_{n=1}^{N} \frac{\text{Uniform}(a_n)}{\pi_c(a_n)} \log p_{\theta,v}(r_n|a_n, x_n). \quad (6)$$

In more detail, counterfactual risk minimization is required because if the collection policy π_c is not uniform then the resulting

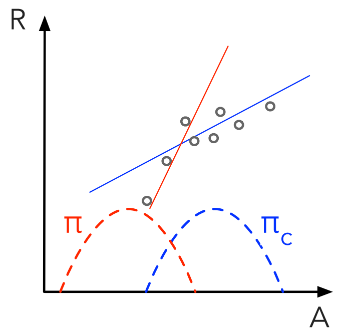


Figure 3: Off-policy training fits a reward function that best fits the input points with respect to the target policy π using input points generated by the collection policy π_c.

objective when training a model does not match the target objective Eq. 5 in expectation. Furthermore, no amount of data will correct for this mismatch. Eq. 6 addresses the problem by modifying the weight of each data point's contribution to the objective function to make it look like an expectation with respect to the uniform policy.

Figure 3 illustrates the problem that CRM addresses.[2] Suppose the goal is to minimize the sum of squared errors with respect to π using data generated from π_c. Simply minimizing the sum of squared errors of the observed data points would result in the blue line. But this is not the answer we are seeking because there are more data points where π_c is higher. This gives that region of action space more importance than other regions during the line fitting. Eq. 6 tells us to downweight points where π_c is high and π is low (and vice versa) before minimizing the sum of squared errors. This results in the steeper red line.

As an example in a recommender systems context, if the deployed recommender is suggesting item j_1 with explanation e_1 to a large number of users, and the same item with a different explanation (j_1, e_2) to a small number of users, then the logged data will reflect these proportions. Training a new model from the logged data will unduly focus on minimizing the error for explanation e_1 at the expense of e_2 even if the latter has substantially more engagement. In a linear discrete model, such as $r^{(1)}$ in Eq. 1 when the context consists of one-hot encodings, there is no trade-off. But any non-linear model, such as a factorization machine or neural network, will sacrifice unbiasedness for generalization.

It follows that we must know the collection policy π_c for each recommendation, referred to as the propensity score in causal analysis, in order to do counterfactual training. We consider how to do this next.

3.2.3 *Exploration-Exploitation Policy & Propensity Scoring.*

We finally define the exploration-exploitation approach and collection policy for propensity scoring in Bart.

When presented with context x and user u, the optimal action under the reward function is $(j^*, e^*) = \arg_{j,e} \max r(j, e, x)$. There are several exploration approaches to select from, most based on variants of Thompson sampling, upper confidence bounds, and epsilon-greedy [22]. We focus on epsilon-greedy for simplicity of implementation in production and of propensity scoring.

A standard epsilon-greedy treatment of the reward function gives equal probability mass to all non-optimal items in the validity set $f(e, x)$ and $(1-\epsilon)$ additional mass to the optimal action (j^*, e^*).[3] The limitation of this approach is that the policy must either exploit (j^*, e^*) or explore the item and explanation simultaneously. But if the contribution an explanation alone makes to engagement is non-zero (i.e., if $\theta_e \neq 0$ in Eq. 1- 4) then this is overly exploratory as there may exist a set of effective explanations that Bart should focus on regardless of the items.[4] Furthermore, exploring over a large number of actions results in small propensity scores. This has the effect of giving the counterfactual objective in Eq.6 high variance, making training unreliable. Finally, in practice, we are sometimes constrained to presenting multiple items within an explanation at once. This most commonly occurs in the shelf interface on websites.

In light of these considerations, Bart uses conditional exploration. Specifically, it decides whether to explore or exploit the items separately from the explanations, all the while keeping the same underlying reward model r that captures the interactions between items, explanations, and the context,

$$\pi_c^{\text{item}}(j \mid x, e) = \begin{cases} (1-\epsilon) + \frac{\epsilon}{|f(e,u)|}, & \text{if } j = j^*, j \in f(e,x) \\ \frac{\epsilon}{|f(e,u)|}, & \text{if } j \neq j^*, j \in f(e,x) \\ 0, & \text{otherwise.} \end{cases}$$
$$\text{where } j^* = \arg_{j_1} \max r(j_1, e, x) \tag{7}$$

$$\pi_c^{\text{expl.}}(e \mid x, j) = \begin{cases} (1-\epsilon) + \frac{\epsilon}{|\mathcal{E}|}, & \text{if } e = e^*, j \in f(e,x) \\ \frac{\epsilon}{|\mathcal{E}|}, & \text{if } e \neq e^*, j \in f(e,x) \\ 0, & \text{otherwise.} \end{cases}$$
$$\text{where } e^* = \arg_{e_1} \max r(j, e_1, x). \tag{8}$$

Items are sampled first from π_c^{item} for all $e_1 \in \mathcal{E}$. Then, conditional on the items chosen, the explanation is sampled from $\pi_c^{\text{expl.}}$. The overall propensity score is therefore $\pi_c = \pi_c^{\text{item}} \pi_c^{\text{expl.}}$. To further address small propensities giving the objective high variance, a combination of normalized importance sample reweighting and propensity score capping is used [7].[5]

3.2.4 *Training and Action Selection in Practice.*

Incorporating all the elements introduced in Sections 3.2.1-3.2.3, we briefly discuss how Bart trains and selects actions in practice. Ideally, the parameters are recalculated from all the data seen up to the N^{th} (latest) observation using Eq. 6. This takes constant time in fully conjugate models such as a Gaussian or Bernoulli model of rewards. However, models that generalize better across contexts (e.g., logistic regression, factorization machine) require at least $O(N)$ training time and N grows by 1 each time a new observation is made. For this reason, Bart perioridcally retrains in batch mode (Eq. 6), e.g., once per day or once per training dataset. Performance improves the shorter this period because the model has seen more data during

[2]The example in Figure 3 is done in continuous space because it is easier to visualize. The same principles carry over to discrete actions.

[3]If there happens to be two or more equal optimal actions the additional probability mass may be split equally between them.

[4]Decreasing the ϵ parameter does not solve the issue because it only changes the proportion of exploration actions, not what the exploration actions look like.

[5]To further reduce variance could have used a doubly robust approach [5] but found that the variance was sufficiently low due to the restricted action space size.

Table 3: Subset of Shelves Considered in Hypothesis Tests

Explanation	# Impressions
Because it's `[day of week]`	140.3K
Inspired by `[user]`'s recent listening	138.4K
Because it's a new release	140.5K
Because `[user]` likes `[genre]`	130.7K
Because it's popular	140.5K
Mood	140.7K
Focus	140.5K

training. However, the effect of additional data for large training sets is not significant. Our estimated performance can be seen as a lower bound on actual performance. Using the latest reward model, the action for each impression is selected using Eqs. 7 and Eq. 8.

4 EMPIRICAL EVALUATION

We use logged feedback data and live production traffic from an online music streaming service to evaluate different versions of Bart against benchmarks. In our experiments, we find that personalizing explanations and recommendations provides a significant increase in estimated user engagement, as estimated by offline and online metrics. We start in Section 4.1 with offline evaluations where we test two hypotheses and find that explanations have a significant impact on user engagement. We perform a set of offline experiments that show that considering pairwise and three-way interactions within the context vector is a much more accurate approach for Bart. In Section 4.2 we describe our experiments in an online A/B test, including the challenges involved in building a production scale test. We find that Bart provides more than a 20% improvement in stream rates compared with statically ordering explanations.

4.1 Offline Evaluation

Offline evaluation uses logged user interaction data with randomized recommendations to check hypotheses about explainable recommendations and to compare different recommendation algorithms. A summary of the offline data collected from a streaming mobile music recommendation service is shown in Table 4.

The recommendations were *safely randomized*, meaning that we pre-select approximately 100 relevant playlists for each user (as defined by the validity function f from Section 3.1) then randomly shuffled the explanation *and* the set of recommendations within each explanation. Relevance was determined by a separate embedding-based model of preference that is not related to the reward model in Bart, similar to the candidate selection stage in Covington et al. [4]. While safe exploration introduces some confounding into the data, it limits the negative impact on user satisfaction that a fully randomized trial would create (e.g., by recommending a genre to a user who strongly dislikes that genre). As we will describe in this section, we addressed this additional confounding in the hypothesis testing by subselecting playlists and explanations.

4.1.1 Testing the Explanation Hypothesis. We first test the null hypothesis that explanations have no effect on user satisfaction. Rejecting the null hypothesis means that users respond differently depending on which explanation is used, and if this is the case, it motivates the learning of personalized explanations for users.

Null Hypothesis 1. *The outcome of whether an explanation results in user stream is independent of the choice of explanation.*

We use a χ^2 test to determine whether the null can be rejected by the offline data at 1% significance. We focus on the top 7 explanations to ensure that there were no population differences between the groups that see each explanation. In doing so, we account for the candidate selection step outlined in Section 4.1. The 7 explanations examined here are listed in Table 3. There are 6 degrees of freedom to the test, giving a threshold of $\chi^2_{1\%,6} = 22.458$. The data gives $\chi^2 = 570.67$. Therefore, we reject the null hypothesis and find that stream behavior is dependent on the choice of explanation for the 7 popular explanations in our data set.

Could the difference in stream probability be due to the fact that different explanations have different valid items? In other words, the fact that explanation A has higher engagement than explanation B could be because users respond better to A or because A has a more engaging set of items to recommend (or both). We investigate whether there is an intrinsic appeal to some shelves for some users with the next null hypothesis.

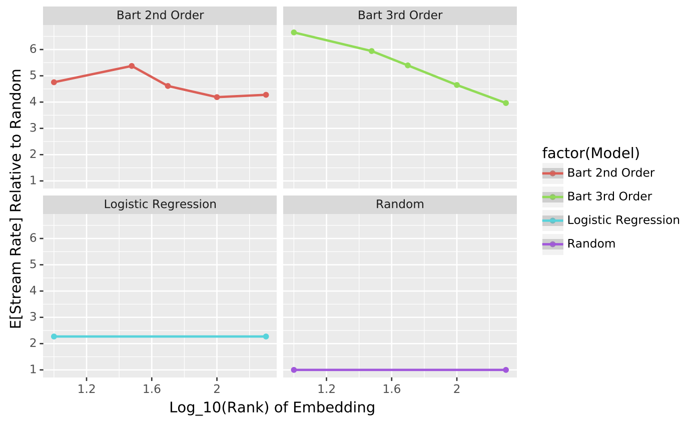
Null Hypothesis 2. *The outcome of whether a user streams from a recommendation is independent of the choice of explanation provided for that recommendation.*

To check Null Hypothesis 2, we focus on the subset of playlists that appear in more than one explanation. There were two sets of explanations that exhibited significant playlist overlap. The first set consisted of set_1 = { *day-of-the-week, inspired-by-recent-listening, because-user-likes-genre, popular* } resulting in a $\chi^2_{3,1\%}$ threshold of 16.266 at 3 degrees of freedom. The second set consisted of set_2 = { *mood, focus* } resulting in a $\chi^2_{1,1\%}$ threshold of 10.828. We set the significance threshold for both tests at 1%.

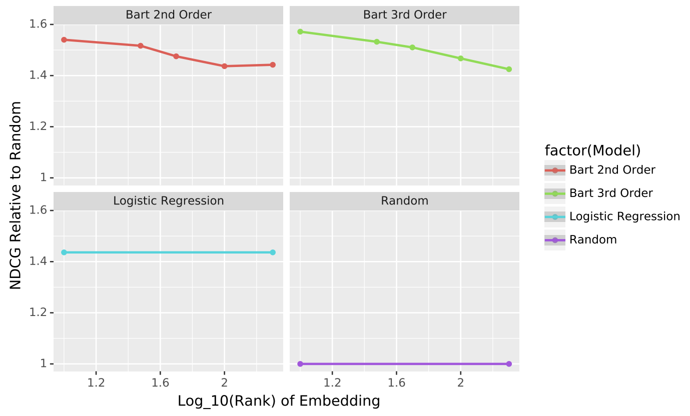
The data indicate $\chi^2 = 126.85$ for set_1 and $\chi^2 = 0.11236$ for set_2, meaning that we can reject the null hypothesis for set_1 but not set_2. Therefore, we know that explaining the same playlist as recommended because it is popular vs. because the user likes a certain genre vs. because it's linked to the day of week vs. because it is similar to what the user has been listening to recently makes a difference to engagement. Specifically, popularity was the weakest driver of engagement while related to recent listening and genre-based explanations were the strongest. [6]

Our failure to reject the hypothesis for set_2 means that we must accept Null Hypothesis 2 for the *mood* and *focus* explanations. This is unsurprising because these are vaguer explanations and weaker calls to action than the other explanations we considered. Also, in practical terms, we only found 6 playlists in common between the two explanations, giving us less data with which to draw conclusions about those explanations.

[6] An alternative reason why recent listening might have stronger engagement, even with the same playlists, is because the recommendations are presented at the right *time*. While this likely contributes to the effect, the evidence for rejecting Null Hypothesis 2 is strong enough that this additional effect does not change our conclusion.



(a) Expected Stream Rate



(b) NDCG

Figure 4: Experimental results on offline music recommendation data show that a model that captures three-way interactions between contextual features outperforms other methods. Logistic and Random have no embeddings and are therefore constant in that dimension.

Having established a basis on which explained recommendations changes the response to recommendations, we next test various versions of Bart and compare to the unpersonalized policy.

4.1.2 Comparative Offline Experiments. We compare the impact that various recommendation methods have on user satisfaction estimated from logged randomized interaction data. We split the logged data into three sets: the training set comprising the first week of the data, the validation set comprising one day, and the test set comprising the next week. A summary of test data is given in Table 4. We used the whole training week but randomly subsampled 1% of the test week so the training data sizes are approximately 100x those of the test set. This gave us a large enough test set for gaining statistical significance in our results. The recommendation methods, metrics, and results are discussed next.

Table 4: Summary of Test Data

Experiment Group	# Impressions	# Users	# Items
Hypothesis testing	970K	140K	140K
Offline experiments	190K	8.6K	9.6K
Online experiments	3.8M	560K	230K

Contextual Features. The contextual features of the reward model used in the offline and online experiments consist of item attributes and user attributes. The item attributes are the playlist identifier, the explanation identifier, and type of playlist (e.g., album, compilation). The user features are the user region, the product and platform of the user's device, a vector representing the normalized genres from the user's listening history, and a vector representing the normalized types of playlist from the user's listening history.

Recommendation Methods. We compare different versions of Bart against baselines. The methods we consider in the offline experiments are,

- **Bart 2nd Order** uses Bart with a 2nd order factorization machine reward function $r^{(2)}$ given in Eq. 3.
- **Bart 3rd Order** uses Bart with a 3rd order factorization machine reward function $r^{(3)}$ given in Eq. 4.
- **Logistic Regression** provides a static ordering of the items and explanations according to their individual controlled popularity defined in Eq. 2.
- **Random** randomly selects items and explanations while respecting the validity function. The quality of random shuffle is lower bounded due to safe exploration (see Section 4.1).

Metrics. The two metrics we use for comparisons are designed to approximate user satisfaction in a deployed recommender system. Metrics are averaged over users in the test and are all reported relative to Random for data privacy issues. The metrics for all bandit-based approaches include the cost of exploration.

- **Expected Stream Rate** The expected stream rate is the offline estimate of the average number of times a user streamed at least one song from a recommended playlist per recommendation. Evaluation on recommendation data can sometimes give misleading results due to confounding, as described in Section 3.2.2. To address this, we use inverse propensity score (IPS) reweighting to estimate the expectation [1]. Issues of high variance are known to be associated with IPS [7]. We did not encounter this problem here due to moderate action space size and no extreme propensities.
- **NDCG** Normalized discounted cumulative gain measures the weighted frequency of streams that a recommendation algorithm ranks near the top of a list. Formally, it is defined as NDCG@K $= \frac{\text{DCG@K}}{\text{IDCG@K}}$, where DCG@K $= \sum_{k=1}^{K} \frac{2^{\text{rel}_k}-1}{\log_2(k+1)}$ and $\text{rel}_k = 0$ or 1 is the measured relevance of the item that a recommender placed at position k, in this case whether a user streamed at least one track from a playlist. IDCG@K is the idealized DCG, where all items are relevant, normalizing the NDCG score. We set $K = 10$ to the number of items that would typically be considered in a recommendation.

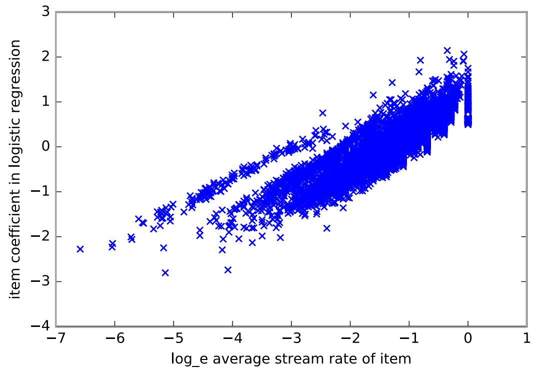


Figure 5: Relationship between item coefficient in logistic regression and item popularity shows that they are closely related but not identical.

Results. Figure 4a and Figure 4b show the expected stream rate and NDCG for the recommendation methods under a range of embedding sizes and order of interactions. We find that Bart with a reward model that considers third order interactions between elements of the context vector performs the best across a variety of embedding sizes and is typically 5 times better than random as measured by stream rate. The reward model that considers second order interactions performs almost as well, but is more sensitive to the size of the embedding and appears to peak at embedding size 30. We see no benefit to increasing the embedding size beyond 30 for any approach because larger embeddings overfit the data.

As described in Section 3.2.1, logistic regression in recommendation provides a static ordering of items and explanations according to controlled popularity. Plotting the learned controlled popularity of items against the true stream rate in Figure 5 tells us the strength of relationship before and after controlling for non-item factors relating to the user and item context. It is interesting to see a large range of variation in average stream rate for any particular coefficient. For example, the set of items with a controlled popularity coefficient = 1 have actual stream rates in the range $[e^{-2}, 1]$. This tells us that Logistic Regression is controlling for strong external factors would otherwise confound the popularity estimate in a static ordering of items and explanations.

4.2 Online A/B Tests

We now evaluate Bart in an online A/B test with live production traffic from an online music recommender service. For this test we introduce the Control benchmark that ranks the explanations statically according to their engagement but uses a default ordering of items. In this section we start by discussing the production challenges related to this test, describe the control experiment and contextual features considered, then discuss the results.

Production Challenges. There are several challenges we faced moving Bart into a production setting for online A/B testing. Firstly, too much exploration has the potential to confuse users. To address this, we limited the exploration rate in our experiments to 10%, giving significant probability mass to all actions in the space while reducing the negative effects of exploration. We also set the causal unit to one day per recommendation. Secondly, prediction has strict latency requirements. This constrained the set of features we can use. Thirdly, the features had to be available in both batch (offline) and real time settings. Some features are hard to make available in real time, such as the set of all artists a user has ever listened to. Other features are hard to make available in batch, such as the last 15 songs listened to. Fourthly, it is imperative that the feature extraction and processing is identical during both training and production. It is easy for inconsistencies to be introduced if these occur in different languages or systems.

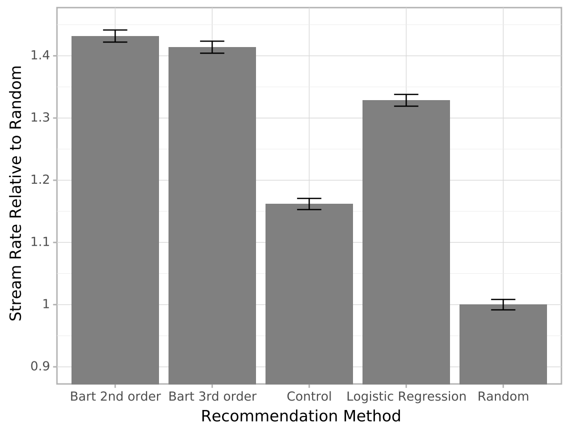


Figure 6: Online results from a music recommender system. Error bars indicate the 95% confidence interval.

Results. The results in Figure 6 mostly agree with the offline experiments from Figure 4. Considering two way and three way interactions with Bart provides a considerable boost over the Logistic Regression, Control, and Random methods. Since the confidence intervals for Bart 2nd Order and Bart 3rd Order overlap, we cannot say that one has a higher stream rate than the other here.

The two sets of experiments with results in Figure 6 and Figure 4 appear to disagree in both the ordering of method performance and magnitude of improvement over Random. The magnitude difference could be due to the fact that the online test breaks the independence assumption of rewards. Specifically, if an action results in a playlist being streamed then this may affect other streaming behavior immediately and possibly in the longer term.

5 CONCLUSIONS & FUTURE WORK

In this paper we introduced Bart, an algorithm for jointly personalizing recommendations and associated explanations for providing more transparent and understandable suggestions to users. Bart is an efficient method for addressing the exploration-exploitation problem in recommendation. Experiments show that explanations affect the way that users respond to recommendations and that Bart significantly outperforms the best static ordering of explanations.

There several open issues to be addressed in future work. Further automating the generation and parameterization of explanations would enable more nuanced personalization. Also, future work can incorporate slate assumptions to better recommend sets of items. Finally, we would like to consider other exploration methodologies for more efficient exploration.

REFERENCES

[1] Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. 2014. Taming the monster: A fast and simple algorithm for contextual bandits. In *International Conference on Machine Learning*. 1638–1646.

[2] Svetlin Bostandjiev, John O’Donovan, and Tobias Höllerer. 2012. TasteWeights: a visual interactive hybrid recommender system. In *Proceedings of the sixth ACM conference on Recommender systems*. ACM, 35–42.

[3] Allison J. B. Chaney, Brandon Stewart, and Barbara Engelhardt. 2017. How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. *arXiv preprint arXiv:1710.11214* (2017).

[4] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for YouTube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 191–198.

[5] Miroslav Dudík, John Langford, and Lihong Li. 2011. Doubly robust policy evaluation and learning. *arXiv preprint arXiv:1103.4601* (2011).

[6] Gerhard Friedrich and Markus Zanker. 2011. A taxonomy for generating explanations in recommender systems. *AI Magazine* 32, 3 (2011), 90–98.

[7] 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.

[8] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM’08. Eighth IEEE International Conference on*. Ieee, 263–272.

[9] Thorsten Joachims, Dayne Freitag, and Tom Mitchell. 1997. Webwatcher: A tour guide for the world wide web. In *International Joint Conference on Artificial Intelligence (IJCAI)*. 770–777.

[10] Thorsten Joachims and Adith Swaminathan. 2016. Tutorial on Counterfactual Evaluation and Learning for Search, Recommendation and Ad Placement. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*. 1199–1201.

[11] Antti Kangasrääsiö, Dorota Glowacka, and Samuel Kaski. 2015. Improving controllability and predictability of interactive recommendation interfaces for exploratory search. In *Proceedings of the 20th international conference on intelligent user interfaces*. ACM, 247–251.

[12] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).

[13] Pigi Kouki, James Schaffer, Jay Pujara, John O’Donovan, and Lise Getoor. 2017. User Preferences for Hybrid Explanations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 84–88.

[14] Branislav Kveton, Csaba Szepesvari, Zheng Wen, and Azin Ashkan. 2015. Cascading bandits: Learning to rank in the cascade model. In *International Conference on Machine Learning (ICML)*. 767–776.

[15] Branislav Kveton, Zheng Wen, Azin Ashkan, Hoda Eydgahi, and Brian Eriksson. 2014. Matroid Bandits: Fast Combinatorial Optimization with Learning. In *Conference on Uncertainty in Artificial Intelligence (UAI)*.

[16] Paul Lamere. 2017. https://twitter.com/plamere/status/822021478170423296. Twitter. (2017).

[17] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web (WWW)*. ACM, 661–670.

[18] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. 2008. Learning diverse rankings with multi-armed bandits. In *International Conference on Machine Learning (ICML)*.

[19] Steffen Rendle. 2010. Factorization machines. In *IEEE 10th International Conference on Data Mining (ICDM)*. IEEE, 995–1000.

[20] Guy Shani, David Heckerman, and Ronen I Brafman. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6, Sep (2005), 1265–1295.

[21] Anongnart Srivihok and Pisit Sukonmanee. 2005. E-commerce intelligent agent: personalization travel support agent using Q Learning. In *Proceedings of the 7th international conference on Electronic commerce*. ACM, 287–292.

[22] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge.

[23] Adith Swaminathan, Akshay Krishnamurthy, Alekh Agarwal, Miro Dudik, John Langford, Damien Jose, and Imed Zitouni. 2017. Off-policy evaluation for slate recommendation. In *Advances in Neural Information Processing Systems*. 3635–3645.

[24] Nava Tintarev and Judith Masthoff. 2007. A survey of explanations in recommender systems. In *IEEE 23rd International Conference on Data Engineering Workshop*. IEEE, 801–810.

[25] Xinxi Wang, Yi Wang, David Hsu, and Ye Wang. 2014. Exploration in interactive personalized music recommendation: a reinforcement learning approach. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 11, 1 (2014), 7.