Interpretable Attribute-based Action-aware Bandits for Within-Session Personalization in E-commerce

Xu Liu[1], Congzhe Su[2], Amey Barapatre[2], Xiaoting Zhao[2], Diane Hu[2], Chu-Cheng Hsieh[2], Jingrui He[3]
[1] Arizona State University, [2] Etsy Inc., [3] University of Illinois at Urbana-Champaign
xliu338@asu.edu, {csu, abarapatre, xzhao, dhu, chsieh}@etsy.com, jingrui.he@gmail.com

Abstract

When shopping online, buyers often express and refine their purchase preferences by exploring different items in the product catalog based on varying attributes*, such as color, size, shape, and material. As such, it is increasingly important for e-commerce ranking systems to quickly learn a buyer's fine-grained preferences and re-rank items based on their most recent activity within the session. In this paper, we propose an Online Personalized Attribute-based Re-ranker (*OPAR*), a light-weight, within-session personalization approach using multi-arm bandits (MAB). As the buyer continues on their shopping mission and interacts with different products in an online shop,* OPAR *learns which attributes the buyer likes and dislikes, forming an interpretable user preference profile and improving re-ranking performance over time, within the same session. By representing each arm in the MAB as an attribute, we reduce the complexity space (compared with modeling preferences at the item level) while offering more fine-grained personalization (compared with modeling preferences at the product category level). We naturally extend this formulation to weight attributes differently in the reward function, depending on how the buyer interacts with the item (e.g. click, add-to-cart, purchase). We train and evaluate* OPAR *on a real-world e-commerce search ranking system and benchmark it against 4 state-of-the-art baselines on 8 datasets and show an improvement in ranking performance across all tasks.*

1 Introduction

When buyers shop online, they are often faced with thousands, if not millions, of products to explore and potentially purchase. In recent years, we've seen a growing interest in industrial applications of ranking systems as they help minimize distractions for the buyer and surface a digestible number of products that are most relevant to their shopping mission. These ranking systems take the form of search or recommendation systems, where products are ranked in descending order of relevance to the buyer [13, 17, 21, 31, 33, 37].

Just as a shopper might browse the aisles of a shop, online shoppers also spend time on a retailer's website searching and clicking on items before they decide what they want to buy. This process is an attempt to refine their purchase intent as they learn more about the product catalog. For example, a buyer might be interested in purchasing a ring; however, they often must click on a number of different rings before they understand possible styles, shapes, colors, and materials that are available. Eventually, the buyer might decide that they have a

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Figure 1: The first two components show a typical 2-stage ranker, where the first-pass narrows down the product catalog to relevant items, while the second-pass performs fine-grained re-ranking to optimize for a business metric. The proposed model, *OPAR*, is responsible for within-session, online personalization that can be effective on its own or as a third-pass ranker on top of a 2-stage ranking system.

preference for an emerald gemstone, with a circular shape, and a gold band. Shifting to looking for a necklace, the buyer must refine their preference again. Often the buyer's preference for attributes like colors and materials changes quickly over the course of one visit. An intelligent ranking system must continually serve content that stays relevant to the buyer's changing preference, a capability we refer to as *within-session personalization*.

Many production ranking systems today have multiple goals to balance: online retailers not only surface content that is *relevant* to the shopper's buying mission (for example, a search query for "wristwatc" must produce wrist watches), but they also aim show content that is likely to improve a business metric (eg. conversion rate, or GMV). In order to balance these goals, many production ranking systems leverage a 2-stage ranking process (Figure 1): the first pass (commonly referred to as *candidate set selection*) narrows hundreds of millions of items from the product catalog down to a few hundred relevant items [14, 21, 36]; the second pass then re-ranks the top few hundred relevant items in a way that optimizes for specific user action (such as a click or purchase) [9, 10, 22, 31]. In order to maximize prediction accuracy, these systems often train on billions of historical data points that may span over the course of months or years and thus cannot react quickly enough to the buyer's changing preference within a shopping visit.

In this paper, we propose an *O*nline *P*ersonalized *A*ttribute-based *R*e-ranker (*OPAR*) that can respond quickly to the changing preferences of a buyer within their immediate shopping session, while still reaping the benefits of a traditional 2-stage system. In the MAB literature, it is common to address this problem by treating each arm in the bandit to represent a single item [37], product category [28, 33] or a context [13, 15, 16]. In contrast, *OPAR* decomposes each product into a descriptive set of attributes (such as its color, texture, material, and shape), and represents each arm as an *attribute*. As the buyer interacts with different products in an online shop, the bandit learns which attributes the buyer likes and dislikes, forming an interpretable user preference profile that is used to re-rank products in real-time in a personalized manner. By representing each arm as an attribute, we reduce the complexity of the space, while allowing more fine-grained personalization within a product category. We naturally extend this formulation to weight attributes differently in the reward function, depending on how the user interacts with that item (e.g. attributes from a clicked item will be weighted less than attributes from an add-to-cart item).

Figure 2: Example of attribute and action-aware re-ranking by *OPAR*. From left to right: (1) shows search results for the query "Ring". User 1 clicked on two gemstone rings (outlined in green), while User 2 adds a diamond ring to their cart (outlined in blue) (2) The attribute of the clicked items are "Crystal", "Gemstone", "Ruby" and "Rose Gold", while the add-to-cart item has the attributes "Diamond", "Engagement", "Oval-Cut" and "14k Gold" (3) On a subsequent search page, *OPAR* re-ranks items based on each user's diverging preferences.

In our example of searching for a ring, we see in Figure 2 that initially, the same 12 items are shown to two different users. While user 1 might click on items that contain the attributes *crystal, gemstone, ruby, rose gold* (outlined in *green*), user 2 might have different preferences and click on items that contain the attributes *diamond, engagement, oval cut, 14K gold* (outlined in blue). At this point, *OPAR* will begin to differentiate the diverging preferences of these two users based on the different attributes that each user has shown interest in. On a subsequent search page, *OPAR* will rank gemstone rings higher for user 1, while user 2 will see diamond rings at the top of the list. Furthermore, because the learned weights of each attribute can be observed for each user, our model is extremely interpretable.

While *OPAR* can be used as a stand-alone algorithm, we find it to be most effective when deployed as a third-pass ranker on top of a traditional two-stage ranking system (see Figure 1). This allows us to leverage the power of traditional 2-pass systems that learn from long-term data aggregated over billions of user and item preferences, while still being nimble enough to personalize a buyer's experience by taking into account their most recent activity.

In the following, we will introduce the proposed model, *OPAR*, and show how we apply it to a search ranking problem on a popular e-commerce platform. Our contributions are as follows:

- **Attribute-level personalization**: *OPAR* performs real-time personalized re-ranking based on user's preferences at the attribute level and reduces the space complexity while offering more fine-grained personalization.

- **Light-weight, online re-ranker**: *OPAR* improves ranking performance with little data and requires us to track a minimal number of variables as arms and can be added on top of the traditional 2-pass ranking systems.

- **Interpretable user preferences**: The learned attribute weights give visibility into attributes that the user likes and dislikes. Top-weighted ones can be used for down-stream personalization tasks.

- **Evaluation on real-world datasets**: *OPAR* is trained and evaluated on real-world e-commerce data and is compared to baselines on 8 datasets from a production e-commerce ranking system. We describe a session-level ranking metric to understand ranking improvements within a session.

2 Related Work

In this section, we summarize the related work from literature and categorize them into two aspects: (1) *Session-based Ranking System*, and (2) *Multi-armed Bandit Ranking System*.

2.1 Within-Session Ranking

The within-session ranking task tries to predict what action the user will take next within the current shopping session, leveraging the temporal nature of their browsing behavior from within the same session [17, 34]. Significant breakthroughs in deep learning (i.e, batch normalization and dropout), have led to its wide adoptions in various communities and applications [35]. In [11], recurrent neural networks (RNNs) were proposed for this within-session ranking task and gained significant attraction given its superior predictive performance for the next-item recommendation. This has been an active research area with various enhancements proposed specifically for predicting short-term user behavior within the same shopping session [11, 12, 24, 27, 34].

Given that a long-term memory models are insufficient to address drift in user interests, [18] proposed a short-term attention priority model to capture users' general (long-term) interest in addition to the users' within-session interest via a short-term memory model based on the recent clicks. In parallel, [17] studied the behavior-intensive neural network for personalized next-item recommendation by considering both users' long-term preference as well as within-session purchase intent. As RNNs have shown and emerged as the powerful technique to model sequential data for this task, [20] argued for an alternative model, inspired by machine translation, by proposing an encoder-decoder neural architecture with an attention mechanism added to capture user session intents and inter session dependencies. In addition to sequential models, [23] leverages graph neural networks by constructing a session graph and then modeling a weighted attention layer when predicting user's preference in session. To tackle uncertainty that arises in a user's within-session behavior, authors in [8] proposed a Matrix Factorization-based attention model to address large-volume and high-velocity session streaming data and [19] handles the missing value issue for the matrix factorization.

Most previous work cited above do not aim for interpretability of its results. In contrast, the model we propose specifically leverages item attributes from the product catalog, resulting in a simple algorithm that learns interpretable user profiles that aid in within-session personalization. The closest related work is [4] that proposes the attribute-aware neural attentive model for the next shopping basket recommendation but does not seem to easily adapt for the real-time scenario due to its complexity.

2.2 Multi-Armed Bandits Ranking System

Requiring a responsive and scalable ranking system that can adapt to the dynamic nature of shifting user preferences (especially in the cold start setting) has led to increasingly wider industry adoption of multi-armed bandit (MAB) in modern day ranking systems. The theoretical foundation and analysis of MABs have been well-studied, with popular approaches include ϵ-greedy [26], Upper Confidence Bounds [2], Thompson sampling [7], EXP3 [3], and others [26]. In the e-commerce [37] setting, the goal is to maximize user satisfaction (i.e., exploitation), while quickly learning (i.e., exploration) users preferences by exploring unseen content.

Hu et al. in [13] proposed to use reinforcement learning to learn an optimal ranking policy that maximizes the expected accumulative rewards in a search session. Yan et al. from [33] built a scalable deep online ranking system (DORS) with MABs as the last pass to dynamically re-rank items based on user real-time feedback and showed significant improvement in both users satisfaction and platform revenue. Furthermore, authors from [25] proposed a multi-armed nearest-neighbor bandit to achieve collaborative filtering for the interactive recommendation, by modeling users as arms and exploring the users' neighborhood. [29] proposed an interactive collaborative topic regression model that infers the clusters of arms via topic models [5] and then utilizes dependent arms for the recommendation.

Figure 3: Top attribute-value pairs for top categories: (1) Jewelry; (2) Clothing; (3) Craft Supplies and Tools, (4) Home and Living.

In this literature, it is common to address this problem by treating each arm in the bandit to represent a single item [37], product category [33] or a context [13, 15, 16]. In contrast, *OPAR* decomposes each product into its descriptive set of attributes (such as its color, texture, material, and shape), represents each arm as an *attribute* and provides great explainability in addition to its performance.

3 Problem Formulation

In this section, we provide definitions for commonly used terms such as sessions and attributes. We then explain our model in two parts: (1) how to represent within-session attribute preferences, and (2) how to re-rank items based on these preferences.

3.1 Definitions

Definition 1: A *session* is a sequence of actions that the buyer takes while engaging with an e-commerce platform in trying to fulfill a shopping mission (e.g. search, click, add-to-cart). The session typically ends when the buyer leaves the site with a purchase or abandons after a significant duration of inactivity (e.g., 30 minutes). Note that we focus on product search here but should be generally applicable to other ranking or recommendation problems.

Let us define a session $S = \{[Q_t, I_t, A_t]\}_{t=1}^{T}$ that is a sequence of T user actions within a session, in which T can vary across sessions. The session starts at $t = 1$ and ends at T with a purchase (or becomes inactive). At each time step, item list $I_t \in R^{M \times 1}$ contains M candidate items to be re-ranked for query Q_t, and then how the user engages with the list of items is represented by A_t:

$$A_t(x_i) = \begin{cases} 0, & \text{no action on } x_i \\ 1, & x_i \text{ is purchased} \\ 2, & x_i \text{ is added to cart} \\ 3, & x_i \text{ is clicked} \end{cases}, \forall x_i \in I_t. \tag{1}$$

Definition 2: An *attribute* is a basic unit (e.g. size, color) that describes the product characteristics of an item. The attributes are determined by taxonomists based on the product category while the value of the attributes (e.g. large, green) are volunteered by the seller, or inferred by machine-learned classifiers. These attribute-value pairs help buyers efficiently navigate through an overwhelmingly large inventory. Thus, each item x_i is represented as the composition of its attributes, with H_{x_i} denoting the total number of attributes associated with $x_i : x_i = \{atr_1, atr_2, \ldots, atr_{H_{x_i}}\}$.

Figure 3 shows four category-specific word clouds of attributes-value pairs exhibited in items from top categories at Etsy, one of the largest e-commerce platform for handmade, vintage, and craft supplies. Some of the most common attributes are universal: size, color, and material. Others are category specific: sleeve length, earring location, and craft type. Lastly, some attributes (e.g. holiday, occasion, recipient) describe how or when the item can be used.

3.2 Problem Statement

Our goal is to (1) formulate each user's within-session preference for product attributes and (2) re-rank a list of candidate items based on the user's inferred within-session preference on item attributes.

Part 1: ***How to formulate users' in-session attribute preferences?***
Input: For session S, (1) item lists $\{I_t\}_{t=1}^{T}$ with each item $x_i = \{atr_{H_{x_i}}\}$ as composition of product attributes; and (2) session-level record of user actions on shown items, $\{A_t\}_{t=1}^{T}$.
Output User's preference $\mathbf{\Theta}$ on attributes as beta-distributed: $\mathbf{\Theta} = \{\theta_{atr_n}\}_{n=1}^{N} \sim \{Beta(\alpha_{atr_n}, \beta_{atr_n})\}_{n=1}^{N}$, where N denotes the total number of attributes encountered in session S.

For a user, we model their within-session preference on an attribute as a latent value $\theta_{atr_n} \in [0, 1]$ denoting the probability that they would like the attribute exhibited in the item. Motivated by Thompson Sampling [1], let θ_{atr_n} be beta-distributed, with $\alpha_{atr_n}, \beta_{atr_n}$ be the two parameters of the distribution. In Section 5.2 we show a method on estimating the parameters of attributes from historical data. From the list of shown items I_t, the user engages on a subset of items (denoted in A_t) to express their preference for item attributes according to $\mathbf{\Theta}$. Given the feedback, we propagate rewards from the user actions to the associated attributes with increments, $\delta_{A_t(x_i)}$, and update the posterior distribution of $\mathbf{\Theta}$, with rewards normalized at x_i by its cardinality (number of associated attributes on that item).

Part 2: ***How to sequentially re-rank I_t based on user preference* $\mathbf{\Theta}$ to optimize in-session personalization?**
Input: At time t, (1) Candidate list of items I_t, and (2) user in-session preference $\mathbf{\Theta}$.
Output: Sequentially learn $f_t : I_t \times \mathbf{\Theta} \to \tilde{I}_t$.

Below we will present the *OPAR* algorithm to address the problem statement discussed here.

4 Proposed Algorithm, *OPAR*

In this section, we present the details of the proposed *OPAR* model and its extension *OPAR*$_w$ which differentiates different user actions.

4.1 Scoring and Re-ranking Item List

Given attribute-level bandits with each arm as an item attribute, we describe below our approach on how we score and re-rank items, motivated by the Thompson Sampling approach on [1].

Let N denote the number of attributes associated with item list I_t. For each attribute in $\{atr_h : atr_h \in x_i, \forall x_i \in I_t\}$, we randomly sample θ_{atr_h} from its corresponding distribution, denoting the probability that the user is interested in the attribute, atr_h, at time t:

$$\theta_{atr_h} \sim Beta(\alpha_{atr_h}, \beta_{atr_h}). \tag{2}$$

Then, each item $x_i \in I_t$ is scored and ranked by:

$$score(x_i) = \sum_{atr_h \in x_i} g(\theta_{atr_h}), \tag{3}$$

where $g(\theta_{atr_h}) = \frac{1}{rank(\theta_{atr_h})}$ is a harnomic function of the index that θ_{atr_h} is ranked among $[\theta_{atr_h}]_{h=1}^{H_{x_i}}$, with a tie-breaker uniformly at random. A larger $score(x_i)$ indicates higher satisfication with item x_i given users' short in-session preference on the attributes. Lastly, we present the user $\tilde{I}_t$, which is reranked list of the items based on $[score(x_i)]_{x_i \in I_t}$.

```
Algorithm 1: OPAR Algo: Re-Ranking & Parameter Update
  Input:
    Given a session $S = \{[Q_t, I_t, A_t]\}_{t=1}^{T}$
    $\{\delta_i\}_{i:\text{ actions}}$: action-aware increments on attribute parameters
    $\gamma$: hyper-parameter to control intensity on negatives
    $\mathcal{U}_t$, $\mathcal{V}_t$: the associated attributes from engaged items; the associated attributes from impressed items
    $|\cdot|_0$: cardinality operator
  for $[Q_t, I_t, A_t] \in S$ do
    (1) Rerank on the Item List $f : I_t \rightarrow \tilde{I}_t$
      sample $s_{atr_h} \sim Beta(\alpha_{atr_h}, \beta_{atr_h}), \forall atr_h \in N_S$
    for $x_i \in I_t$ do
      Given $x_i = \{atr_h\}_{h=1}^{H_{x_i}}$ as associated attributes in $x_i$, set $score(x_i) = \sum_{atr_h \in x_i} g(s_{atr_h})$
    end
    $\tilde{I}_t = sorted([score(x_i)]_{x_i \in I_t})$
    (2) Update attribute parameters given $A_t$
    Let $\mathcal{U}_t = \cup\{atr_h : \forall atr_h \in x_i\, if\, A_t(x_i) \neq 0, \forall x_i \in I_t\}$
    Let $\mathcal{V}_t = \cup\{atr_h : \forall atr_h \in x_i \forall x_i \in I_t\}$
    for $x_i \in I_t$ do
      if $A_t(x_i) \neq 0$, item $x_i$ has positive actions then
        $\alpha_{atr_h} += \delta_{A_t(x_i)} \times \{1 - Exp\,(-|\mathcal{U}_t|_0)\}, \forall atr_h \in x_i$
      else if $A_t(x_i) = 0$, no action on item $x_i$ then
        $\beta_{atr_h} += \delta_{A_t(x_i)} \times \{1 - Exp\,(-\gamma|\mathcal{V}_t \backslash \mathcal{U}_t|_0)\}, \forall atr_h \in x_i$
      end
    end
  end
  Output: All re-ranking results $[\tilde{I}_t]_{t=1}^{T}$
```

4.2 Attribute Parameter Updates

With the feedback gathered from the user action A_t, we do the following updates for the attribute parameters. Let $\mathcal{U}_t$ denote the set of attributes associated from items with positive actions (i.e., click, add-to-cart, purchase), and $\mathcal{V}_t$ be union of all attributes exist in $x_i \in I_t$:

$$\mathcal{U}_t = \cup\{atr_h : \forall atr_h \in x_i \text{ if } A_t(x_i) \neq 0,\ \forall x_i \in I_t\},\ \mathcal{V}_t = \cup\{atr_h : \forall atr_h \in x_i,\ \forall x_i \in I_t\}.$$

For a given atr_h, let $\tilde{\mathcal{Y}}_{t,atr_h}$ and $\tilde{\mathcal{Z}}_{t,atr_h}$ denote the set of items associated with positive user action and no-action, respectively,

$$\tilde{\mathcal{Y}}_{t,atr_h} = \{x_i \in I_t : atr_h \in x_i \text{ and } atr_h \in \mathcal{U}_t\},\ \tilde{\mathcal{Z}}_{t,atr_h} = \{x_i \in I_t : atr_h \in x_i \text{ and } atr_h \in \mathcal{V}_t \backslash \mathcal{U}_t\}.$$

Then, the Beta distribution of each attribute is updated as follows:

$$\begin{aligned} \alpha_{atr_h} &+= \sum_{\tilde{\mathcal{Y}}_{t,atr_h}} \delta_{A_t(x_i)} \left(1 - e^{-|\mathcal{U}_t|_0}\right), \forall atr_h \in \mathcal{U}_t, \\ \beta_{atr_h} &+= \sum_{\tilde{\mathcal{Z}}_{t,atr_h}} \delta_{A_t(x_i)} \left(1 - e^{-\gamma|\mathcal{V}_t \backslash \mathcal{U}_t|_0}\right), \forall atr_h \in \mathcal{V}_t \backslash \mathcal{U}_t, \end{aligned} \tag{4}$$

where $|\cdot|_0$ denotes the cardinality operator and γ controls intensity on implicit no-actions.

Table 1: Etsy Real-world Session-based Dataset Over 3 weeks

ID	Category	|Session (User)|	|Query|	|Item|	|Attributes|	|Actions|
1	**Clothing**	4642	46091	1100040	2495	58932
2	**Home & Living**	9073	103959	2282542	2455	134416
3	**Paper & Party Supplies**	4419	35132	691919	1666	55037
4	**Craft Supplies & Tools**	10913	123662	2536492	2799	171363
5	**Accessories**	5813	38215	897533	2419	49342
6	**Electronics & Accessories**	1638	10505	216860	1302	14354
7	**Jewelry**	5585	67507	1530285	2266	79874
8	**Overall Category**	26442	474594	9295453	3363	624882

4.3 *OPAR* algorithm Procedure

In summary, given a session $S = \{[Q_t, I_t, A_t]\}_{t=1}^{T}$, *OPAR* can be summarized with the following steps, with the pseudo code of *OPAR*$_w$ shown in Algorithm 1.

1. Initialize attribute dictionary $atrDic \in R^{N \times 2}$, which contains N pairs of parameters for attributes, where each row of $atrDic$ denotes the Beta distribution parameter set $(\alpha_{atr}, \beta_{atr})$ for a given attribute. Different initializations have been experimented, including uniform, random or estimated based on held-out historical datasets (shown in Section 5.2).

2. At time t, we score each item $x_i \in I_t$ based on Eq. (3): it first aggregates over the associated attribute preferences sampled in Eq. (2), and then re-rank items based on scores in Eq. (3) and present as $\tilde{I}_t$. More details in Section 4.1.

3. At time t, we receive the obervation A_t on I_t, and then update the distribution of all attributes associated with item x_i in the *atrDic* based on the Eq. (4) described in Section 4.2.

   OPAR: attribute-based bandits with *equal* action-weighting for actions in $\{\text{click}, \text{add-to-cart}, \text{purchase}\}$. This means that for positive actions, $\delta_{\text{click}} = \delta_{\text{add-to-cart}} = \delta_{\text{purchase}}$.

   OPAR$_w$: extend *OPAR* to weight action-aware updates as follows, $\delta_{\text{click}} \neq \delta_{\text{add-to-cart}} \neq \delta_{\text{purchase}}$, and hypertune them.

4. We iterate step (2) and (3) until the end of the session.

5 Experiments

In this section, we show how *OPAR* performs on a real-world e-commerce ranking system and benchmark it against 4 baselines on 8 datasets. While *OPAR* can be applied to any content that requires re-ranking, we specifically chose to train, evaluate, and analyze the model performance on a search ranking system, as the explicit search queries issued by a user shows higher purchase intent, allowing us to better evaluate *OPAR*'s ranking and interpretation capabilities. Our experimentation seeks to answer the following questions:
Experiment #1: What is the ranking performance of the proposed *OPAR* model? *(Answered in subsection 5.4.1)*
Experiment #2: How does *OPAR* perform as an action-aware model? *(Answered in subsection 5.4.2)*
Experiment #3: How does *OPAR* help to understand users' short-term, in-session shopping preference? *(Answered in subsection 5.4.3)*

5.1 Data Collection

The dataset is collected and sampled from a month of user search logs at Etsy, one of the largest e-commerce platforms for handmade, vintage items, and craft supplies. To avoid bot traffic, filters are added to include sessions with at least 10 search events (i.e., queries, browses, clicks, add-to-carts) and at least one *purchase* to focus on sessions with strong shopping missions. Using an existing query classifier, we predict the most probable category (e.g. jewelry, home and living) associated with the first query of each session, and then bucket the entire session into one of 7 categories. This helps us understand shopping behaviors within each category. Table 1 shows statistics of each data set, representing nearly 500k search queries from 26k sessions and 620k user actions combined on nearly ten million items, with cardinalities computed within each dataset. We do not perform the evaluation on existing public datasets, because (to the best of our best knowledge) there is no existing dataset that includes all meta-data needed for our study (e.g. query, item attribute, user interaction logs).

5.2 Experimental Set-up

We split each of the 8 datasets into 2 parts (with sessions ordered chronologically). The first two-thirds of the data is a **held-out dataset**. Because we are focused on online learning, using only within-session data, the held-out dataset is mainly used for estimating the parameters of the Beta distributions, $\{(\alpha_{atr}, \beta_{atr})\}_{\forall atr}$, and to aggregate attribute counts associated with engaged items to determine attribute popularity, powering the "Atr-POP" algorithm in Section 5.3. The remaining data is the **testing dataset**, on which we report re-ranking performance for *OPAR* and other baseline algorithms on in Table 2.

While *OPAR* can function as a stand-alone ranking algorithm, we evaluate OPAR (as well as other baselines) on top of an existing 2-pass ranking system (as described in Figure 1). More formally, each session in the testing dataset, $S = \{[Q_t, I_t, A_t]\}_{t=1}^{T}$ contains a sequential list of query content Q_t, a candidate set I_t of items to be re-ranked, and logged user actions A_t on I_t (e.g. click, purchase). In our experiments, I_t is a truncated list of the top 48 items returned by an existing 2-pass ranker, indicating that this list comprises of the most relevant items to the query. As we will see in experimental results, applying *OPAR* adds an effective layer of attribute-based personalization in real-time that was not feasible with the underlying system. In order to simulate an online environment, only within-session user interactions leading up to the current time step are used for ranking predictions.

5.3 Evaluation Metrics and Baselines

Below, we describe the offline metrics we use to evaluate *OPAR* on the testing dataset, as well as the baselines we benchmark.

5.3.1 Evaluation Metrics

Following the general ranking metric Normalized Discounted Cumulative Gain (NDCG) [30], we propose a set of session-level ranking metrics to evaluate our model.

1. *Click-NDCG*: For each query Q_t issued in S that has at least one click in A_t (i.e, clicks as relevances), *click-NDCG*$_t$ measures the re-ranking performance of the item list $\tilde{I}_t$ (after re-ranking I_t) shown to the user at t. For all timestamp with at least a click, we first compute stepwise sequential re-ranking performance *click-NDCG*$_t$ as:
$$click\text{-}NDCG_t = click\text{-}DCG_t / IDCG_t, \forall t = 1, ...T, \tag{5}$$
and *click-NDCG* of a session S is the average of *click-NDCG*$_t$ over events that have at least one click:
$$click\text{-}NDCG = \text{Average}(click\text{-}NDCG_t). \tag{6}$$

		Over All Category							
		LambdaMART	Atr-KNN	Atr-POP	GRU4Rec	OPAR	OPAR$_w$	LambdaMART	Atr-KNN
Purchase NDCG	@4	0.1795	0.0130	0.0749	0.0618	0.2994	**0.3042**	0.1948	0.0103
	@12	0.2629	0.0412	0.1323	0.1425	0.3505	**0.3607**	0.2670	0.0348
	@24	0.3162	0.1260	0.2112	0.2018	0.3718	**0.3900**	0.3019	0.0090
	@48	0.3724	0.2554	0.2861	0.2518	0.4512	**0.4578**	0.3774	0.2462
Click NDCG	@4	0.1459	0.0816	0.0705	0.0701	0.3120	**0.3158**	0.1328	0.0067
	@12	0.2265	0.1456	0.1264	0.1354	0.3213	**0.3229**	0.2137	0.0228
	@24	0.2955	0.2157	0.2021	0.1922	0.3318	**0.3489**	0.2821	0.0658
	@48	0.3815	0.3245	0.2813	0.2689	0.4047	**0.4051**	0.3711	0.2309

		Home & Living						Pape	
		LambdaMART	Atr-KNN	Atr-POP	GRU4Rec	OPAR	OPAR$_w$	LambdaMART	Atr-KNN
Purchase NDCG	@4	0.1755	0.0131	0.0649	0.0571	0.2920	**0.2952**	0.1822	0.0010
	@12	0.2670	0.0396	0.1226	0.1281	0.3391	**0.3436**	0.2667	0.0406
	@24	0.3218	0.0936	0.2066	0.1752	0.3838	**0.3879**	0.3276	0.1297
	@48	0.3874	0.2543	0.2789	0.2164	0.4462	**0.4491**	0.3876	0.2550
Click NDCG	@4	0.1481	0.0054	0.0601	0.0944	0.3201	**0.3219**	0.1585	0.0052
	@12	0.2294	0.0213	0.1175	0.1416	0.3244	**0.3256**	0.2394	0.0247
	@24	0.2978	0.0598	0.1973	0.1843	0.3485	**0.3491**	0.3103	0.0644
	@48	0.3835	0.2278	0.2746	0.2288	0.4032	**0.4086**	0.3911	0.2306

		Craft Supplies & Tools							
		LambdaMART	Atr-KNN	Atr-POP	GRU4Rec	OPAR	OPAR$_w$	LambdaMART	Atr-KNN
Purchase NDCG	@4	0.1912	0.0135	0.0739	0.0741	0.3101	**0.3268**	0.1954	0.0251
	@12	0.2735	0.0407	0.1296	0.1125	0.3673	**0.3781**	0.2828	0.0741
	@24	0.3272	0.1208	0.1970	0.1644	0.4084	**0.4188**	0.3324	0.1406
	@48	0.3844	0.2577	0.2820	0.2214	0.4366	**0.4750**	0.3869	0.2693
Click NDCG	@4	0.1458	0.0055	0.0749	0.0994	0.3118	**0.3166**	0.1502	0.0105
	@12	0.2262	0.0513	0.1290	0.1279	0.3241	**0.3293**	0.2324	0.0439
	@24	0.2955	0.2042	0.1953	0.1935	**0.3525**	0.3521	0.3006	0.1091
	@48	0.3811	0.2278	0.2815	0.2277	**0.4080**	0.4078	0.3848	0.2391

		Electronics & Accessories							
		LambdaMART	Atr-KNN	Atr-POP	GRU4Rec	OPAR	OPAR$_w$	LambdaMART	Atr-KNN
Purchase NDCG	@4	0.2136	0.0501	0.0715	0.0814	0.2847	**0.2995**	0.1661	0.0060
	@12	0.3014	0.1109	0.1546	0.1223	0.3386	**0.3782**	0.2534	0.0766
	@24	0.3519	0.0176	0.2652	0.1674	**0.4257**	0.4152	0.3087	0.1470
	@48	0.4060	0.2965	0.2981	0.2416	0.4516	**0.4656**	0.3814	0.2460
Click NDCG	@4	0.1530	0.0267	0.0805	0.0641	**0.2074**	0.2051	0.0701	0.0027
	@12	0.2324	0.0703	0.1580	0.0939	0.2487	**0.2622**	0.1314	0.0106
	@24	0.3029	0.1410	0.2657	0.1345	**0.3158**	0.3120	0.1989	0.1276
	@48	0.3880	0.2560	0.3026	0.1667	0.3978	**0.4078**	0.3119	0.2192

Table 2: Re-ranking performance comparison on over all data sets (top-left) and 7 category-specific data sets.

2. *Purchase-NDCG*: Following the above methodology, we compute the session-level re-ranking performance limit to search events with attributed purchases. A session on a shopping site is defined as a sequence of events ending with a purchase or a significant duration of inactivity. Given that, *Purchase-NDCG* given a session is essentially *purchase-NDCG$_T$*.

For each re-ranking algorithm reported in Table 2, we compute *Click-NDCG* @k and *Purchase-NDCG*@k for $k = \{4, 12, 24, 48\}$ by averaging *click-NDCG$_s$* @k and *purchase-NDCG$_s$*@k given session s over all sessions in each dataset. Note that k is a multiple of 4 as that this shopping site displays 4 items per row on desktops.

5.3.2 Baselines

We compared *OPAR*'s ranking performance with 4 state-of-the-art baselines:

1. LambdaMART [32] is the boosted tree version of LambdaRank [6], which introduces the use of gradient boosted decision trees for solving a ranking task and won Track 1 of the 2010 Yahoo! Learning To Rank Challenge. A personalized search re-ranker is trained based on long-term user historical data to optimize for the user's purchasability on an item given the query issued and the user's historical preference.

Table 3: Multiple Purchase Intents within One Session

	Timestamp	Query	Query Taxonomy		
1st Purchase Intent	0	’flower girl basket’	paper and party supplies	(NO ACTION)	Bro
	1-4	’flower girl basket wedding’	paper and party supplies	(CLICK)	’Pri ’We
	5-9	’flower girl basket beach wedding’	paper and party supplies	(CLICK)	’Pri ’We
	10-11	’two flower girl and one pillow’	paper and party supplies		Bro
				Purchase Intent Change	
2nd Purchase Intent	12-15	’hat for beach wedding’	clothing.women_clothing	(CLICK)	’Pri
	16-22	’turquoise petals’	accesories	(CLICK)	’Pri
	23	’bride hair decoration beach theme’	clothing.women_clothing	(NO ACTION)	Bro
Final Purchase	24	’turquoise petals’	accesories	(PURCHASE)	’Pri

2. Atr-KNN is derived from Item-KNN [11]. Each item is presented by n-hot-encoding of associated attributes with n being the cardinality of all attributes. That is, its i^{th} entry equals to 1 if the referred attribute presents in the item, otherwise 0. Items in the list I_{t+1} are re-ranked based on their euclidean-distance from the last engaged item(s) in I_t. Note that the items $x_i \in I_t$ with no-action has no impact on this re-ranking.

3. Atr-POP reranks the candidate set, I_t, of items based on the attributes’ popularity estimated with held-out historical records. This baseline is one of the most common solutions derived from [11] given its simplicity and efficacy.

4. GRU4Rec [11] applies recurrent neural networks (RNN) on short session-based data of clicked items to achieve session-based next-item recommendation. Each session is encoded as a 1-of-N vector, in which the i^{th} entry is 1 if the corresponding item is clicked else 0, with N denoting the number of items. While the user’s consecutive clicks on items are used in the next item prediction, it is attribute-agnostic.

While it is common for each arm in the bandits to represent a single item or product category, we skip it as a baseline here as this would incur higher exploration cost with potential latency bottleneck when scaling up to an inventory of hundred millions of items and also lose interpretability of product attributes.

5.4 Experimental Results

In this section, we describe experimentation results for evaluating three kinds of performance: (1) ranking performance, (2) impact of differentiating between user action types, and (3) interpretability.

5.4.1 Overall Re-ranking Performance

Table 2 shows experiment results of our model (*OPARs*) against 4 baselines described in Section 5.3. The results can be categorized into two parts: (1) performance on the aggregated datasets over all categories (top-left); and (2) performance on each of the 7 category-specific datasets, representing different shopping missions and behaviors across categories (i.e, “Clothing”, “Home & Living”). Across all 8 datasets for the re-ranking task, *OPAR*$_w$ outperform against all 4 baselines, including LambdaMART, Atr-KNN, Atr-POP, and GRU4Rec in both purchase-NDCG and click-NDCG.

For the overall dataset (top-left), *OPAR*$_w$ shows over 6% lift in click-NDCG@48 compared to the best baseline, and over 20% increase in purchase-NDCG@48. Similar results are observed in each category-specific re-ranking. For k, the best improvement for *OPAR*$_w$ is achived at $k = 4$, ordering by @4 >> @12 >> @24 >> @48. With attribute-based bandits, interactive feedbacks from the in-session user actions, even just fewer clicks, efficiency propagate rewards to associated attributes and quickly learns preferred attributes that matter the most to the user, thus optimize user purchase intent.

Figure 4: In-session *OPAR* re-ranking performance.

5.4.2 Effectiveness of Action-aware MABs

To explore users' in-session activity with different types of actions (i.e, click, add-to-cart), we run experiments with the action-aware bandit model, with *OPAR*$_w$ hypertuned rewards from clicks vs add-to-carts, to differentiate *types* of user actions. The results in Table 2 are reported from a tuned model that assigns larger weights to *clicks* than *add-to-carts*, with an intuition that there is a high topical drift observed in the user's browsing intent after items are added to carts. As shown in Table 2, collectively *OPAR*$_w$ outperforms *OPAR* by 1.6% and 1.1% in purchase NDCG@4 and click NDCG@4, respectively. When segmenting by categories, *OPAR*$_w$ also outperforms *OPAR* in almost all categories, except *Electronics* & *Accessories* and *Craft Supplies* & *Tools* on purchase NDCG.

5.4.3 Interpretability of Within-Session Shopping Mission

It is often observed that a user exhibits multiple purchase intents with diverse preferences within a session. Table 3 presents a record of a user's in-session activities. Figure 4 (top) shows the sequential improvement of *OPAR* in session-level click-NDCG over time compared to the baseline, and Figure 4 (bottom) shows how *OPAR* captures user's preference, θ_{atr_h}, on 5 attributes over time. The "Engaged Attributes" column in Table 3 maps out all attributes associated with the clicked items for the corresponding query.

As shown in Table 3, the user is interested in three categories as his/her purchase intents: first in "paper & party supplies", then drift to "women clothing" and "accessories", and lastly converted in "accessories" with a *purchase*. After the browsing period from timestamp $t = 0$ with no user actions, β_{atr} for the attributes associated with the browsing-only items are incremented while no attributes have been updated with positive rewards for the given user. *OPAR* launches from a lower click-NDCG at the beginning, while obtains better re-ranking performance compared with baseline by learning that the user is interested in white prime color and is looking for the wedding occasion theme by the end of $t = 4$. From then *OPAR* outperforms the baseline in click NDCG while activated more attributes related to wedding themes in beach and tropical and expanded to floral crafting type and blue for prime color. The re-ranking performance continues to improve from $t = 5, .., 9$ as more items related to these attribute themes are discovered.

Starting from $t = 12$, the user starts to explore the 2nd categorical purchase intent, pivoting from "paper and party supplies" to "clothing" and "accessories". However, the latest activated attributes based on the engaged items on the first set of shopping queries still relevant. The user has a consistent preference in attributes, such as "Prime Color: Blue", "Occasion: Wedding", and "Wedding theme: Fairytale & princess" as she is searching for a "hat for beach wedding" and/or "bride hair decoration beach theme". Thus, for the second purchase intent starting at $t = 12$, we observe a high jump start in *OPAR*'s click NDCG at $t = 12$ comparing to the first intent at $t = 1$ and the metric continues to stepwise improve. As demonstrated in Figure 4 (bottom), "wedding theme" and "primary color: blue" are the top two performant attributes that *OPAR* learned and identified over time.

6 Conclusion

This paper proposes an interpretable *Online Personalized Attributed-based Re-ranker (OPAR)* as a light-weight third-pass, followed by the normal 2-stage ranking process, to personalize a buyer's in-session experience based on product attributes. Given the important presence of attributes in the product category with its simplicity in explainability, we propose attribute-based multi-armed bandits to quickly learn the buyer's fine-grained preferences and re-rank items based on the recent activities within the session to achieve in-session personalization. We then extend the reward function of the attribute-based bandits to weight based on the type of actions the buyer interacts with the item (i.e, click, add-to-cart, purchase). Lastly, we train and evaluate *OPAR* on the real-word e-commerce search ranking system, and show its superior performance against the baselines across multiples datasets. For future works, we could consider bias correction (i.e, position) in parameter updates to reduce self reinforcing, and model interactions between query and attributes to capture user preferences on attributes beyond engaged items.

References

[1] S. Agrawal, N. Goyal. Analysis of thompson sampling for the multi-armed bandit problem. Conference on learning theory, 2012.

[2] P. Auer, N. Cesa-Bianchi, P. Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. Machine Learning, 2002.

[3] P. Auer, N. Cesa-Bianchi, Y. Freund, R-E. Schapire. The Nonstochastic Multiarmed Bandit Problem. Society for Industrial and Applied Mathematics, 2003.

[4] T. Bai, Ting, J-Y. Nie, W-X. Zhao, Y. Zhu, P. Du, J-R. Wen, Ji-Rong. An Attribute-Aware Neural Attentive Model for Next Basket Recommendation. The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18), 2018.

[5] D-M. Blei, A-Y. Ng, M.I. Jordan. Latent Dirichlet Allocation. Journal of Machine Learning Research, 2003.

[6] C-J. Burges, R. Ragno, Q-V. Le. Learning to rank with nonsmooth cost functions. Advances in Neural Information Processing Systems, 2007.

[7] C. Olivier, L. Li. An Empirical Evaluation of Thompson Sampling. Advances in Neural Information Processing Systems 24, 2011.

[8] L. Guo, H. Yin, Q. Wang, T. Chen, A. Zhou, N. Quoc Viet Hung. Streaming Session-Based Recommendation. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2019.

[9] R. Guo, X. Zhao, A. Henderson, L. Hong, H. Liu. Debiasing Grid-based Product Search in E-commerce. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '20), 2020.

[10] M. Haldar, P. Ramanathan, T. Sax, M. Abdool, L. Zhang, A. Mansawala, S. Yang, B. Turnbull, J. Liao. Improving Deep Learning for Airbnb Search. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '20), 2020.

[11] B. Hidasi, A. Karatzoglou, L. Baltrunas, D. Tikk. Session-based Recommendations with Recurrent Neural Networks. arXiv: 1511.06939, 2015.

[12] L. Hu, L. Cao, S. Wang, G. Xu, J. Cao, Z. Gu. Diversifying Personalized Recommendation with User-Session Context. Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17), 2017.

[13] Y. Hu, Q. Da, A. Zeng, Y. Yu, Y. Xu. Reinforcement Learning to Rank in E-Commerce Search Engine: Formalization, Analysis, and Application. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '18), 2018.

[14] J-T. Huang, A. Sharma, S. Sun, L. Xia, D. Zhang, P. Pronin, J. Padmanabhan, G. Ottaviano, L. Yang. Embedding-Based Retrieval in Facebook Search. Proceddings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '20), 2020.

[15] L. Shuai, K. Purushottam. Context-Aware Bandits. arXiv: 1510.03164, 2015.

[16] S. Li, B. Wang, S. Zhang, W. Chen, Wei. Contextual Combinatorial Cascading Bandits. Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48 (ICML'16), 2016.

[17] Z. Li, H. Zhao, Q. Liu, Z. Huang, T. Mei, E. Chen. Learning from History and Present: Next-Item Recommendation via Discriminatively Exploiting User Behaviors. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2018.

[18] Q. Liu, Y. Zeng, R. Mokhosi, H. Zhang. STAMP: Short-Term Attention/Memory Priority Model for Session-Based Recommendation. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2018.

[19] X. Liu, J. He, S. Duddy, L. O'Sullivan, Liz. Convolution-consistent collective matrix completion. Proceedings of the 28th ACM international conference on information and knowledge management, p:2209–2212, 2019.

[20] P. Loyola, C. Liu, Y. Hirate. Modeling User Session and Intent with an Attention-Based Encoder-Decoder Architecture. Proceedings of the Eleventh ACM Conference on Recommender Systems, 2017.

[21] P. Nigam, Y. Song, V. Mohan, V. Lakshman, W. Ding, A. Shingavi, H. Teo, H. Gu, B. Yin. Semantic Product Search. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2019.

[22] P. Pobrotyn, T. Bartczak, M. Synowiec, R. Białobrzeski, J. Bojar. Context-Aware Learning to Rank with Self-Attention. arXiv:2005.10084, 2020.

[23] R. Qiu, J. Li, Z. Huang, H. Yin. Rethinking the Item Order in Session-Based Recommendation with Graph Neural Networks. Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM '19), 2019.

[24] M. Quadrana, A. Karatzoglou, B. Hidasi, P. Cremonesi. Personalizing Session-Based Recommendations with Hierarchical Recurrent Neural Networks. Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys '17), 2017.

[25] J. Sanz-Cruzado, P. Castells, E. López. A Simple Multi-Armed Nearest-Neighbor Bandit for Interactive Recommendation. RecSys, 2019.

[26] R-S. Sutton, A-G. Barto, Andrew G. Reinforcement learning: An introduction. MIT press, 2018.

[27] Y-K. Tan, X. Xu, Y. Liu. Improved Recurrent Neural Networks for Session-Based Recommendations. Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS'16), 2016.

[28] C-H. Teo, H. Nassif, D. Hill, S. Srinivasan, M. Goodman, V. Mohan, S-V-N. Vishwanathan. Adaptive, Personalized Diversity for Visual Discovery. Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16), p:35–38, 2016.

[29] Q. Wang, C. Zeng, W. Zhou, T. Li, S-S. Iyengar, L. Shwartz, G-Y. Grabarnik. Online Interactive Collaborative Filtering Using Multi-Armed Bandit with Dependent Arms. IEEE Transactions on Knowledge and Data Engineering, 2019.

[30] Y. Wang, L. Wang, Y. Li, D. He, W. Chen, T-Y. Liu. A theoretical analysis of NDCG ranking measures. COLT: Proceedings of the 26th annual conference on learning theory, volume 8, page 6, 2013.

[31] L. Wu, D. Hu, L. Hong, H. Liu. Turning clicks into purchases: Revenue optimization for product search in e-commerce. The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018.

[32] Q. Wu, C. Burges, S. JC and K-M. Svore, J. Gao. Adapting boosting for information retrieval measures. Information Retrieval Journey, 2010.

[33] Y. Yan, Z. Liu, M. Zhao, W. Guo, W-P. Yan, Y. Bao. A practical deep online ranking system in e-commerce recommendation. Springer: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, p:186–201, 2018.

[34] F. Yu, Q. Liu, S. Wu, L. Wang, T. Tan, Tieniu. A Dynamic Recurrent Model for Next Basket Recommendation. Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '16), 2016.

[35] S. Zhang, L. Yao, A. Sun, Y. Tay. Deep Learning Based Recommender System: A Survey and New Perspectives. ACM Comput. Surv., 2019.

[36] X. Zhao, R. Louca, D. Hu, L. Hong. The Difference Between a Click and a Cart-Add: Learning Interaction-Specific Embeddings. Companion Proceedings of the Web Conference, 2020.

[37] Y. Zhao, Y-H. Zhou, M. Ou, H. Xu, N. Li. Maximizing Cumulative User Engagement in Sequential Recommendation: An Online Optimization Perspective. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2020.