

Bulletin of the Technical Committee on

Data Engineering

June 2021 Vol. 45 No. 1



IEEE Computer Society

Editorial Board

Editor-in-Chief

Haixun Wang
Instacart
50 Beale Suite
San Francisco, CA, 94107
haixun.wang@instacart.com

Associate Editors

Bing Yin
Amazon.com
Palo Alto
California, USA

Distribution

Brookes Little
IEEE Computer Society
10662 Los Vaqueros Circle
Los Alamitos, CA 90720
eblittle@computer.org

The TC on Data Engineering

Membership in the TC on Data Engineering is open to all current members of the IEEE Computer Society who are interested in database systems. The TCDE web page is <http://tab.computer.org/tcde/index.html>.

The Data Engineering Bulletin

The Bulletin of the Technical Committee on Data Engineering is published quarterly and is distributed to all TC members. Its scope includes the design, implementation, modelling, theory and application of database systems and their technology.

Letters, conference information, and news should be sent to the Editor-in-Chief. Papers for each issue are solicited by and should be sent to the Associate Editor responsible for the issue.

Opinions expressed in contributions are those of the authors and do not necessarily reflect the positions of the TC on Data Engineering, the IEEE Computer Society, or the authors' organizations.

The Data Engineering Bulletin web site is at http://tab.computer.org/tcde/bull_about.html.

TCDE Executive Committee

Chair

Erich J. Neuhold
University of Vienna

Executive Vice-Chair

Karl Aberer
EPFL

Executive Vice-Chair

Thomas Risse
Goethe University Frankfurt

Vice Chair

Malu Castellanos
Teradata Aster

Vice Chair

Xiaofang Zhou
The University of Queensland

Editor-in-Chief of Data Engineering Bulletin

Haixun Wang
Instacart

Awards Program Coordinator

Amr El Abbadi
University of California, Santa Barbara

Chair Awards Committee

Johannes Gehrke
Microsoft Research

Membership Promotion

Guoliang Li
Tsinghua University

TCDE Archives

Wookey Lee
INHA University

Advisor

Masaru Kitsuregawa
The University of Tokyo

Advisor

Kyu-Young Whang
KAIST

SIGMOD and VLDB Endowment Liaison

Ihab Ilyas
University of Waterloo

Letter from the Editor-in-Chief

The March issue of the Data Engineering Bulletin focuses on the intricate interplay as well as a significant gap between data management and machine learning when it comes to supporting real-life business applications.

The opinion piece of this issue features a group of distinguished researchers and their assessment and prognosis of machine learning's current and future roles in building database systems. Besides highlighting several specific potentials and challenges such as using machine learning to optimize database indices and query optimization, the article also gives a great overview of how databases, data analytics and machine learning, system and infrastructure, work together to support today's business needs. It is clear that the business needs, the volume, velocity, and variety of the data, the latency and throughput requirements have evolved dramatically and in consequence, data management systems must adapt. The opinion pieces described four disruptive forces underneath the evolution, which are likely to influence future data systems.

Our associate editor Sebastian Schelter put together the current issue—Data Validation for Machine Learning Models and Applications—that consists of six papers from leading researchers in industry and academia. The papers focus on data validation, which is a critical component in end-to-end machine learning pipelines that many business applications rely on.

Haixun Wang
Instacart

Letter from the Special Issue Editor

Software applications that learn from data using machine learning (ML) are being deployed in increasing numbers in the real world. Designing and operating such applications introduces novel challenges, which are very different from the challenges encountered in traditional data processing scenarios. ML applications in the real world exhibit a much higher complexity than “text book” ML scenarios (e.g., training a classifier on a pre-existing dataset). They do not only have to learn a model, but must define and execute a whole ML pipeline, which includes data preprocessing operations such as data cleaning, standardisation and feature extraction in addition to learning the model, as well as methods for hyperparameter selection and model evaluation. Such ML pipelines are typically deployed in systems for end-to-end machine learning, which require the integration and validation of raw input data from various input sources, as well as infrastructure for deploying and serving the trained models. The system must also manage the lifecycle of data and models in such scenarios, as new (and potentially changing) input data has to be continuously processed, and the corresponding ML models have to be retrained and managed accordingly. The majority of these challenges have only recently begun to attract the attention of the data management community. A major obstacle is that the behavior of ML-based systems heavily depends on the consumed input data, which can rapidly change, for example due to changed user behavior or due to errors in external sources that produce the inputs. This area represents a gap between the data management and ML communities: research in ML mostly focuses on learning algorithms, and research in data management is mostly concerned with data processing and integration. In this issue, we focus on this gap in data validation for machine learning, and provide perspectives from both the academic and industrial research communities to learn about the state of the art, open problems and to uncover interesting research directions for the future.

The first paper presents *A Data Quality-Driven View of MLOps* and demonstrates how different aspects of data quality propagate through various stages of machine learning development. It connects data quality to the downstream machine learning process, an approach that is also taken by our second paper, which argues that we should move *From Cleaning before ML to Cleaning for ML*. The authors propose an end-to-end approach to take the entire application’s semantics and user goals into account when cleaning data, instead of performing the cleaning operations in an isolated manner beforehand.

The next two papers on *Validating Data and Models in Continuous ML pipelines* and *Automated Data Validation in Machine Learning Systems* from Google and Amazon provide us with an industry perspective on the area in the focus of this issue. The first paper describes tools developed at Google for the analysis and validation of two of the most important types of artifacts: Datasets and Models. These tools (which are part of the Tensorflow Extended Platform) are currently deployed in production at Google and other large organizations, and are heavily inspired by well-known principles of data-management systems. The second paper from Amazon reviews some of the solutions developed to validate data at the various stages of a data pipeline in modern ML applications, discusses to what extent these solutions are being used in practice, and outlines research directions for the automation of data validation.

The subsequent paper on *Enhancing the Interactivity of Dataframe Queries by Leveraging Think Time* focuses on the highly exploratory and iterative nature of data validation in the early stages of an ML application, where data scientists start with a limited understanding of the data content and quality, and perform data validation through incremental trial-and-error. The final paper of this issue on *Responsible AI Challenges in End-to-end Machine Learning* completes the view on data validation for machine learning by connecting it with pressing issues from the area of responsible data management.

Working on this issue has been a privilege for me, and I would like to thank the authors for their contributions.

Bing Yin
Amazon.com, California

utf8]inputenc noend]algpseudocode

Copyright 2021 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Using Product Meta Information For Bias Removal In E-Commerce Grid Search

Apoorva Balyan
Walmart Global Tech India
Apoorva.Balyan@walmart.com

Atul Singh
Walmart Global Tech India
Atul.Singh@walmart.com

Praveen Reddy Suram
Walmart Global Tech India
Praveen.Suram@walmart.com

Deepak Arora
Walmart Global Tech India
Deepak.arora1@walmart.com

Varun Srivastava
Walmart Global Tech India
varun.srivastava@walmart.com

Abstract

In e-commerce, product search plays a crucial role in helping customers discover and purchase products. Most IR algorithms focus on creating a relationship between product and customer intent. These techniques stand on two pillars of information primarily- first, what information the seller provides about a product i.e description, title, taxonomy etc, and second, the implicit feedback data that is collected from the search logs which comprises of millions of user activity events. This data is inherently biased in nature as the user activity is not only dependent on the quality of query-item match but also on how probable the user is to observe that item in the first place. In the IR theory and past research efforts, discounting based on position in evaluation methods have been introduced to address this bias. However, these bias reduction methods cannot be applied efficiently to e-commerce cases because of the difference in the search results displayed to the user. The user behaviour in the list view search differs from that in the grid view search and is composed of three major factors- (1) middle bias (2) slower decay (3) row skipping. There have been efforts to model the user attention probability in a grid-based search. However, these efforts have not been considered item meta-information hence, we propose a method to incorporate the item attributes in the user attention estimation model. The idea is to add a non-positional aspect in the propensity model of the user behaviour patterns. This is based on a simple argument that the user attention is not only dependent on the row and column of the item(aka position) but also on the features of product tile. These features can be shipping promise, price, reviews, ratings etc. Some of these feature tags have been designed to attract the attention of customers. Through various experiments on Walmart search logs data, We show that the proposed framework outperforms the debias baseline algorithms. The results reflect better on how different taxonomies, product categories can impact the user behaviour in grid-based product search.

1 Introduction

There is an inherent difference between traditional search engines and e-commerce or product search engines. In traditional search engines like Google, the search results are displayed in a list view whereas in the e-commerce

search engines like www.walmart.com, the results are displayed in a grid view. The difference in the display of search results can impact the customer's attention and behaviour. Researchers in a previous study [?] were able to show that in the Image Search Engine Result Pages (SERPs), the user's attention can be defined by three factors: (1) User's attention within a row is more in the middle, so there is a bias towards the products in the middle (2) Users tend to skip rows while scrolling on a SERP (3) The decay of user attention is slower in a grid view compared to list view, where the decay happens drastically. This idea was further applied in the e-commerce search [?] and developed an inverse propensity bias model to correct the learning to rank algorithm. While position/presentation of the products on a SERP has been considered in the modelling of the bias, the impact of the product's meta-information on the user attention has still not been addressed.

Intuitively, the SERPs of the e-commerce search engines are not only different in the presentation but also in the fact that e-commerce is a marketplace which has a direct monetary benefit for the sellers maintaining the product information on the page. Thus, the information present for each product tile such as 'Top Picks', 'Reviews' and 'Ratings' would have a very strong impact on the customers navigating this web space to get the best product suited for their requirements. Figure 1 and 2 shows the difference in results displayed in list vs grid view on walmart web and application.

The impact of these qualitative product features on bias would be different for different categories of products. For example : On a category of Milk with clear intent, the new customer might not be exploring so much on the view presented whilst on a category of clothing, the user's attention would be influenced by these product tile tags as discussed above. Hence, we extend the framework to model the product meta-information like Ratings, Reviews, Shipping Promise in the propensity model. Incorporating these features into the model will show the impact on the features displayed on the product tile. To model these features, we use a scoring algorithm to assign a score to each product and this score is modelled with the propensity scoring model. Our aim is to learn a ranker and evaluate the impact of the model using an evaluation metric. We also look at the different product taxonomies like Entertainment, Fashion etc because the customer's intent is different in each product taxonomy. We organize the rest of the paper in the following way: Section 2 contains related work on grid based e-commerce search, Section 3 contains our proposed framework with details on how the item feature scores are calculated and how the propensity score for modelling is derived, Section 4 mentions the experimental settings, evaluation metric and the results, along with the dataset description, and Section 5 presents the conclusion and the future work. Each section is further divided into subsections.

2 RELATED WORK

We will start with a brief introduction of the background and knowledge on the prior research work for analyzing the differences in the user behavior patterns in the list view vs the grid view search, and how this behaviour pattern is used in the propensity score modelling with a brief introduction to LambdaMART ranker.

2.1 GRID SEARCH USER BEHAVIOUR PATTERNS

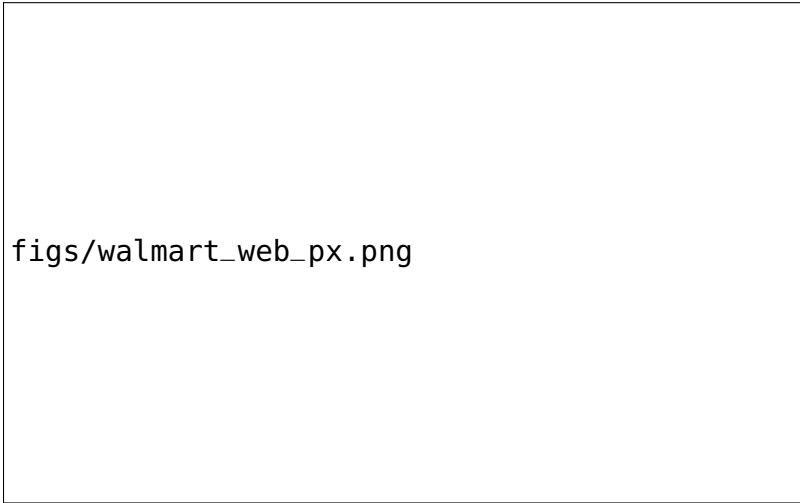
Products in an e-commerce are placed in different view as compared to the majority of search engines that show the list view for the results. The different placement of products leads to a change in the interaction mechanisms of items. Previously, a number of studies have been done on the user behaviour on image or grid search engines [? ? ?]. All these studies focused on comparing the user interaction behaviour in grid search setting to list view display setting by using search logs and eye tracking methods. Important differences in user behaviour, user being more exploratory and more time spent by the user are some of the findings that were observed. We list the major findings in grid based search as follows:

(a) Middle Bias: There exists a "middle position bias" in the user's examination behaviour. Customers allocate their attention more to the items placed in middle positions in a row. The probability of the user examining



figs/walmart_app_px.jpg

Figure 1: List view on Walmart App



figs/walmart_web_px.png

Figure 2: Grid view on Walmart web

items placed on left and right corner in a grid search is less compared to the items in the center. In [?], the researchers consider the dwell time as the factor which is defined as the examination duration of an image and they confirm the middle bias phenomenon. Also, the user behaviour at each position in a row follows normal distribution. Researchers have not used this phenomenon in Learning to rank models of e-commerce yet, and we do not consider it in this research because in the dataset that we use, the number of columns is less than 4 or 5, and our hypothesis is that there are other details in e-commerce search that draw the user attention. So, the middle bias phenomenon is not very effective here.

(b) Row Skipping: The customers do not examine each and every product in a grid from top to bottom, they tend to skip some rows. Users ignore particular rows and directly jump to some distant rows. The probability of the users skipping the first row is very low as compared to the other rows. In [?], row skipping bias in the user behavior was considered in getting an Unbiased Learning to rank model. The probability of the user skipping a

particular row was considered as a bias factor when calculating the lambda gradient for a query. The observation was that the row skipping models performed better in taxonomies where user intent was very clear and users were looking for specific set of items. In such cases, users were probable to skip the rows.

(c) Slower Decay: In a list-view web SERP, the attention of the customer decays in monotonically decreasing fashion and is also dramatic but in grid-view, the decay is much slower. The user behavior in the grid search is more exploratory and there is more browsing as there are a lot of items displayed in a page and users don't have to go to next page to view the next set of items as in the list-view. This leads to the user spending more time in the grid search, concentrating on the items, and thus the decay is slow. In [?], the slower decay user behaviour bias was considered in training unbiased learning to rank models.

These findings are incorporated and Normalized Discounted Cumulative Gain(NDCG) evaluation method is also changed based on grid view search user behaviour.

2.2 Click Models

A lot of prior work has been done on-click modelling in the web search by extracting useful information from search logs of any search engine, mainly the click data. Cascade click model is one of the widely adopted technique in list based search that takes multiple types of user feed backs into account and models the user behavior as a sequence of actions[? ?]. There are different types of biases in the click-models, mainly the position bias. Learning from biased data without taking them into consideration leads to a less effective ranking function. In another study, unbiased learning to rank model was introduced where position bias was considered. Inverse propensity weighting is a well known technique adopted to address any kind of bias and is used for unbiased learning [?]. Most of the work in this area assumes that the propensity scores are present in the logs and they study how to reduce the model variance while remaining unbiased. The unbiased learning-to-rank framework is different, because in that, the propensity is not explicitly logged and is buried in the user behavior data implicitly. Unbiased learning to rank does debiasing of the click data and uses it to train the ranker. Unbiased LambdaMART framework [?] was introduced for training an unbiased ranker by jointly estimating the biases at click positions and the biases at non-click positions.

2.3 LambdaMART Ranker

LambdaMART is a widely used Learning to rank algorithm to solve ranking problems in search engines. LambdaMART [? ? ?] uses gradient boosting and the gradient function of the loss function is called lambda function. The minimization of the objective function is performed using the lambda function on the training dataset. In LambdaMART, the lambda gradient of any document is calculated as

$$\lambda_i = \sum_{j:(d_i, d_j)} \lambda_{ij} - \sum_{j:(d_j, d_i)} \lambda_{ji} \quad (1)$$

λ_{ij} is defined as following :

$$\lambda_{ij} = \frac{-2}{1 + \exp(2(f(x_i) - f(x_j)))} |\Delta_{ij}|$$

where λ_{ij} is the lambda gradient defined on a pair of documents d_i and d_j , σ is a constant, $f(x_i)$ and $f(x_j)$ are the scores of the two documents given by LambdaMART, Δ_{ij} denotes the difference between NDCG scores if documents d_i and d_j are swapped in the ranking list.

3 Item Propensity Scoring Model

We will start with a brief introduction of the background and knowledge on propensity estimation and joint optimization within the learning to rank models. Then we provide description of the changes to the proposed

propensity scoring model using item features.

3.1 Estimating Propensity using Product Meta Information

We hypothesize that the user is not only influenced by the presentation of search query results but also with certain attributes or tags that have been designed to get customer's attention by sellers.

We consider four such attributes about an item which can impact user attention:

(1) **Two Day Shipping:** In the fast paced world that we are in currently, every user wants their favorite products to be delivered at the earliest. Users tend to compare delivery options with other online competitors and place their order accordingly. Thus, to get user attention on items that are eligible for super fast delivery, a message is shown on product tile that this item is eligible for delivery within 2 days. Fig 3,4 shows a two day shipping tag for products in the result set.

(2) **Ratings:** Ratings are given by previous users to the products on any e-commerce platform to indicate how satisfied or dissatisfied previous customers were for that product. This attributes tells what their experience was like and how they rate that experience on a scale of 1 to 5. Fig 3,4 shows the rating tag for products in the result set.

(3) **Reviews:** Reviews is the number of reviews a product has received till date. Reviews is a factor in fetching the rating score for an item. Users are attentive towards number of reviews as well along with rating, as an item with a high rating and very low number of user reviews is highly biased. Products with significant number of reviews gains user trust for that products rating.

(4) **Relative Price:** Price is an important information displayed to the user on the product tile. A set of expensive items for a query may not attract user attention and user may not look in details with much attention. This will counter the slower decay factor. With relative price score, we try to get the expensive and cheap products for a query that is displayed to user. This feature is the relative score among price of all products fetched in the recall set for a particular query.

Based on the above features, we estimate a score and experiment with it in the propensity model. In scoring method, We assign a binary value in the scoring functions for 2-day shipping, ratings and reviews.

According to the formulation:

$$S(I) = \gamma \sum_{i=0}^3 K_i \quad (2)$$

where $K_i \in [-1, 1]$ represents binary vote of the item property.

How to assign the vote for score estimation of an item? To assign a vote for scoring, we look at the quantiles for item features in the data. The threshold for assigning a positive vote or a negative vote is decided using quantile values and then the final score is estimated by the summation of all the votes. The scoring algorithm is mentioned in Algorithm 1.

For example:

If ratings \geq 75th quantile-value, we assign a +1 score to the item or If ratings \leq 25th quantile-value, we assign a -1 score to the item etc. In this similar fashion, we compute a score for each item based on these features.

[H]

- 1: Initialize $score_i$ as 0
- 2: **if** $twoDayShipping_i > 0.0$ **then**
- 3: $score_i \leftarrow score_i + 1$
- 4: **end if**
- 5: **if** $rating_i \geq 75thQuantileValue$ **then**
- 6: $score_i \leftarrow score_i + 1$
- 7: **end if**

figs/4-col-min.png

Figure 3: SERP with 4-column view with product meta information like Ratings, Reviews, Two day shipping

figs/5-col-min.png

Figure 4: SERP with 5-column view with product meta information like Ratings, Reviews, Two day shipping

```

8: if  $rating_i \leq 25thQuantileValue$  then
9:    $score_i \leftarrow score_i - 1$ 
10: end if
11: if  $reviews_i \geq 75thQuantileValue$  then
12:    $score_i \leftarrow score_i + 1$ 
13: end if
14: if  $reviews_i \leq 25thQuantileValue$  then
15:    $score_i \leftarrow score_i - 1$ 
16: end if
17:  $score_i$ 

```

3.2 Changing the Slower Decay Model

As mentioned before, in a SERP with a grid view the user attention decreases more slowly when compared to the results in a list view. According to the earlier work [?], the probability of examination of an item at position, i is estimated as:

$$P(o^i = 1) = \prod_{j=0}^{i-1} \min(\beta^{row(j)} * \alpha), 1.0) \quad (3)$$

where $row(i)$ is the row-number for the item at position i , α is likeness of customer browsing the next item, β models the customer's patience in grid view. As we can see, the probability does not account for item meta-information shown on the item tile in SERP. If an user finds any item attractive in SERP, the user tends to spend more time on it and the probability of examination is more. There are multiple details shown on search page that may attract user attention viz. image quality or attractiveness, user ratings of that particular item, number of reviews on it, special offer tag, price of the product or the delivery date of the product. These factors will also contribute to the probability of user examining any particular product in Grid based SERP. Considering this, the

decay factor will change accordingly. Based on this hypothesis, we model the estimated score above into the probability of examination at position, i as:

$$P(o^i = 1) = \prod_{j=0}^{i-1} \min(\beta^{\text{row}(j)+S(I)} * \alpha), 1.0) \quad (4)$$

We train a ranker f based on loss function L with the propensity score of slower decay model computed using α , β and γ hyper parameters. We use LambdaMART with the ranker, gradient boosting trees as explained before. Following on the same lines, we calculate the lambda gradients by applying inverse propensity scoring. Hence, the lambda gradient for k^{th} product can be re-written as:

$$\frac{\delta L}{\delta x_k} = \lambda_k = \sum_q \left(\sum_{y_q^i = k \cap (i,j) \in I_q} \frac{\lambda_{ij}}{P(o^i)} - \sum_{y_q^j = k \cap (j,i) \in I_q} \frac{\lambda_{ij}}{P(o^j)} \right) \quad (5)$$

where i, j are the positions on the SERP, x_q is the feature vector for query-product pair, y_q^i means that the k^{th} product is at position, i and λ_{ij} is defined as following :

$$\lambda_{ij} = \frac{-2}{1 + \exp(2(f(x_q^i) - f(x_q^j)))} |\Delta_{ij}|$$

Δ_{ij} is the value difference in the NDCG evaluation metric if i and j are to be swapped.

Relative Price Feature : Price is another information displayed on the product tile in an E-commerce search. Most of the users have a budget in mind when they visit to shop any product. If ,for a particular query, most of the items in search results are expensive compared to the range user is looking for, the user may not spend much time on that query viewing the results thus the decay will be comparatively faster. Among the set of items displayed, if there is a relatively cheaper product that also has other attractive tile meta-information, the probability of examining it will be higher. As a part of this work, we consider relative price feature separately to observe the impact of the price on the user behaviour and to check if there is any price bias . Relative price calculation is done as follows:

$$\text{relp}(i) = 1.0 - k * \text{abs}(\text{func}(\log(x/\text{mean_price}))) \quad (6)$$

where $\text{relp}(i)$ is the relative price score for an item i and x is the price of that particular item , k is a constant and mean_price is the mean of prices of all items in the recall set of the query, abs is the absolute function to get the absolute value, func can be any activation function[?] like \tanh or sigmoid . For example , \tanh is defined as

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

Sigmoid is defined as

$$\text{sig}(x) = \frac{1}{1 + e^{-x}}$$

Thus, in our dataset, we have the relative price score for all query-item pairs. Based on this, we redefine the probability of examination at position i in a GRID based SERP as

$$P(o^i = 1) = \prod_{j=0}^{i-1} \min(\beta^{\text{row}(j)+S(I)+\eta*\text{relp}(i)} * \alpha), 1.0) \quad (7)$$

where η is the hyper-parameter introduced to tune the relative price feature in the slower decay model.

For training slower decay model including price feature, we calculate the lambda gradient in the gradient function of LambdaMART as above in (7).

4 Experiments

In this section, we conduct a series of experiments to compare our proposed method with the baseline methods for the removal of bias in grid-based e-commerce search. First, we explain our experimental settings, the datasets that we used, how we prepared them, and the metrics we used to evaluate the models. We then report the results obtained in our unbiased learning to rank setting.

4.1 Datasets

In this subsection, we provide a brief overview of the search log datasets that we used for our experiments. We used the data from the search logs of e-commerce website at Walmart, an American multinational retail corporation with a significant online presence. In particular, we use the data from three product taxonomies viz. Fashion, Entertainment (ENT), and Every Day Living (EDL), and by considering them for capturing and understanding the user behaviour and we eventually obtain four datasets, one for each category. Our datasets included queries, items, positions, and item features. The item features include raw features like item title match, item popularity, partial match score, along with engagement features like click rates, ‘add to cart’ rates, order rates and attribute features like the number of reviews, ratings, the relative price of the products among the recall set, and shipping promise.

4.2 Experimental Settings

In this subsection, we describe the experimental setup. The most common way to evaluate the learning to rank model is by its performance on a holdout set that has unseen data. We perform experiments on each product taxonomy, as mentioned in the subsection above. In our datasets, we perform a random split into a training set (80%), validation set (10%), and test set (10%). We carry out a fast grid search to obtain the best hyper-parameter values for the models. To keep the probability value in a reasonable range, we search alpha in $\{0.8, 0.85, 0.9, 0.95\}$, beta in $\{1.05, 1.1, 1.15, 1.2\}$, and gamma in $\{0.001, 0.003, 0.005, 0.007, 0.01, 0.03, 0.05, 0.07, 0.1\}$. For the LAMBDAMART ranker, we search for the learning rate in $\{0.01, 0.05, 0.1\}$, the maximum depth in $\{3, 5, 7\}$, and the minimum data in leaf in $\{20, 30, 40, 50, 60\}$. We also tune for boosting fraction and feature fraction values in the range 0.7, 0.8, 0.9, 1.0 and 0.8, 0.9, 1.0 respectively. The other parameters are the default settings of Unbiased LambdaMART, while we keep the settings consistent for all the baseline and feature models. We consider the slower decay model[25] as the baseline method. Here, we optimize for the position 10 and consider the metric on the validation set to get the optimal parameter values.

4.3 Evaluation Metrics

Here, we explain the evaluation metric considered for the experiment. We consider the traditional evaluation method of Information Retrieval Systems, Normalized Discounted Cumulative Gain (NDCG) as the evaluation metric [? ? ?]. Experiments are performed in offline settings and we obtain the NDCG metric at different

Taxonomy	α	β	γ
Fashion	0.8	1.05	0.01
EDL	0.95	1.15	0.001
ENT	0.9	1.15	0.03

Table 1: Tuned hyperparameters for item propensity scoring model for different taxonomies

Taxonomy	Framework	NDCG@1	NDCG@3	NDCG@5	NDCG@10
Fashion (w/o price)	Benchmark Slower Decay	0.408	0.486	0.535	0.584
	Proposed framework	0.419	0.492	0.539	0.590
Fashion (with price)	Benchmark Slower Decay	0.408	0.486	0.534	0.583
	Proposed framework(with price)	0.420	0.494	0.540	0.590
Every Day Living	Benchmark Slower Decay	0.553	0.627	0.669	0.711
	Proposed framework	0.553	0.628	0.669	0.711
Entertainment	Benchmark Slower Decay	0.503	0.580	0.624	0.669
	Proposed framework	0.503	0.581	0.624	0.670

Table 2: Experimental Results on different taxonomies using held-out dataset.

positions to compare the proposed framework with the baseline method.

$$NDCG_k = \frac{DCG_k}{IDCG_k} \quad (8)$$

where

$$IDCG_k = \sum_{i=1}^{|REL|} \frac{2^{rel_i} - 1}{\log_2(i + 1)}$$

Here, $IDCG@K$ is the normalizer and $|REL|$ is the list of documents ordered by engagement in the set up to position k . To explain the slower decay of user attention we consider $K = 1, 3, 5, 10$ for the NDCGs.

On the held-out dataset of Walmart Search logs, we calculate the prediction scores of our models. Using the prediction labels, we apply this evaluation method to get the metrics at different positions across different categories.

4.4 Experimental Results

In this section, we provide an overview of the experimental results that we obtain. We share insights on: (1) how e-commerce search results get better with the proposed framework, and (2) how the user behaviour varies across the product taxonomies. We mention the results in Table 2 and summarize our observations as follows:

- Our proposed method for slower decay using product tile information outperforms the baseline method in Fashion vertical by significant margin. This clearly shows the effectiveness of our proposed framework and assumption that user attention decay is slower when user is attracted by any of the item features.
- Performance of our proposed framework in Fashion affirms our initial assumption that users are more inclined on item tile information when shopping for clothing accessories and decay is much slower in such a case as user looks for detailed information.
- The proposed framework of using item attributes in slower decay probability computation shows improvement in Entertainment and Every Day living category too compared to the baseline method of slower decay. Performance improvement is not much compared to Fashion, and this may be because the users have a specific intent when they shop in such categories.
- Adding relative price feature in slower decay probability computation for fashion gave slight improvement over the proposed framework model without price feature. This suggests that price is not a major factor in the users' attempt to click on that product and view the details.

We train separate models for different taxonomies to show that the user behaviour patterns differ across categories and the users react differently to the product tile features.

Hyper Parameter Values: Here, we report the values that we obtain for the hyper parameters (α , β , γ) and that achieve the optimal performance for the proposed framework. For the Fashion category, the proposed slower decay model with $\alpha = 0.8$, $\beta = 1.05$ and $\gamma = 0.01$ outperforms the others. Similarly, for the Every Day Living category, the proposed slower decay model with $\alpha = 0.95$, $\beta = 1.15$ and $\gamma = 0.001$ works best. $\alpha = 0.9$, $\beta = 1.15$ and $\gamma = 0.03$ are the optimal hyper-parameter values for the Entertainment category with the proposed slower decay framework. We experimented on Fashion with the price feature and $\alpha = 0.95$, $\beta = 1.15$, $\gamma = 0.05$, $\eta = 0.01$ are the optimal hyper-parameter values for our dataset.

5 Conclusion and Future work

In this paper, we tried to understand the impact of meta-information on the products displayed on the SERPs for user attention. We extended the solution of debiasing the grid product search by modelling the meta-information of the products, like rating, reviews, shipping promise, price in the propensity model. We showed through extensive experimentation how the different product taxonomies have different user behaviour.

As part of future work, we plan to extend this work to incorporate more product tile features and also use the image quality feature to model the user attention in Grid-based E-commerce search engines. We plan to try out different scoring methods as well for tile features score calculation and model middle bias, row skipping behaviour based on product tile features.

References

- [1] K. Järvelin and J. Kekäläinen. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.*, 10.1145/582415.582418, 2002.
- [2] C. JC Burges. From RankNet to LambdaRank to LambdaMART: An Overview. <https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambda-rank-to-lambdamart-an-overview/>, MSR-TR-2010-82, 2010.
- [3] Z. Cao, T. Qin, T.Y. Liu, M.F. Tsai and H. Li. Learning to Rank: From Pairwise Approach to Listwise Approach. *Proceedings of the 24th International Conference on Machine Learning*, 10.1145/1273496.1273513, 2007.
- [4] Z. Hu, Y. Wang, Q. Peng and H. Li. Unbiased LambdaMART: An Unbiased Pairwise Learning-to-Rank Algorithm. *The World Wide Web Conference*, 10.1145/3308558.3313447, 2019.
- [5] X. Xie, J. Mao, Y. Liu, M. de Rijke, Y. Shao, Z.-Ye, M. Zhang, S. Ma. Grid-Based Evaluation Metrics for Web Image Search. *The World Wide Web Conference*, 10.1145/3308558.3313514, 2019.
- [6] R. Guo, X. Zhao, A. Henderson, L. Hong and H. Liu. Debiasing Grid-Based Product Search in E-Commerce. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 10.1145/3394486.3403336, 2020.
- [7] X. Wang, N. Golbandi, M. Bendersky, D. Metzler and M.-Najork. Position Bias Estimation for Unbiased Learning to Rank in Personal Search. *Association for Computing Machinery*, 10.1145/3159652.3159732, 2018.
- [8] B. Geng, L. Yang, C. Xu, X.S. Hua and S.-Li. The Role of Attractiveness in Web Image Search. *Proceedings of the 19th ACM International Conference on Multimedia*, 10.1145/2072298.2072308, 2011.
- [9] X. Xie, J. Mao, Y. Liu, M. de Rijke, H. Chen, M. Zhang, S. Ma. Preference-Based Evaluation Metrics for Web Image Search. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 10.1145/3397271.3401146, 2020.
- [10] N. Craswell, O. Zoeter, M. Taylor and B. Ramsey. An Experimental Comparison of Click Position-Bias Models. *Proceedings of the 2008 International Conference on Web Search and Data Mining*, 10.1145/1341531.1341545, 2008.
- [11] S. K. K. Santu, P. Sondhi and C. Zhai. On Application of Learning to Rank for E-Commerce Search. *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 10.1145/3077136.3080838, 2017.

- [12] Sham. S. Discounted Cumulative Gain. <https://machinelearningmedium.com/2017/07/24/discounted-cumulative-gain/>, 2017.
- [13] Chandekar. Pranay. Evaluate your Recommendation Engine using NDCG. [tps://towardsdatascience.com/evaluate-your-recommendation-engine-using-ndcg-759a851452d1](https://towardsdatascience.com/evaluate-your-recommendation-engine-using-ndcg-759a851452d1), 2020.
- [14] A. Sharma V. Understanding Activation Functions in Neural Networks. <https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0>, 2017.

utf8]inputenc noend]algpseudocode

Copyright 2021 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Using Product Meta Information For Bias Removal In E-Commerce Grid Search

Apoorva Balyan
Walmart Global Tech India
Apoorva.Balyan@walmart.com

Atul Singh
Walmart Global Tech India
Atul.Singh@walmart.com

Praveen Reddy Suram
Walmart Global Tech India
Praveen.Suram@walmart.com

Deepak Arora
Walmart Global Tech India
Deepak.arora1@walmart.com

Varun Srivastava
Walmart Global Tech India
varun.srivastava@walmart.com

Abstract

In e-commerce, product search plays a crucial role in helping customers discover and purchase products. Most IR algorithms focus on creating a relationship between product and customer intent. These techniques stand on two pillars of information primarily- first, what information the seller provides about a product i.e description, title, taxonomy etc, and second, the implicit feedback data that is collected from the search logs which comprises of millions of user activity events. This data is inherently biased in nature as the user activity is not only dependent on the quality of query-item match but also on how probable the user is to observe that item in the first place. In the IR theory and past research efforts, discounting based on position in evaluation methods have been introduced to address this bias. However, these bias reduction methods cannot be applied efficiently to e-commerce cases because of the difference in the search results displayed to the user. The user behaviour in the list view search differs from that in the grid view search and is composed of three major factors- (1) middle bias (2) slower decay (3) row skipping. There have been efforts to model the user attention probability in a grid-based search. However, these efforts have not been considered item meta-information hence, we propose a method to incorporate the item attributes in the user attention estimation model. The idea is to add a non-positional aspect in the propensity model of the user behaviour patterns. This is based on a simple argument that the user attention is not only dependent on the row and column of the item(aka position) but also on the features of product tile. These features can be shipping promise, price, reviews, ratings etc. Some of these feature tags have been designed to attract the attention of customers. Through various experiments on Walmart search logs data, We show that the proposed framework outperforms the debias baseline algorithms. The results reflect better on how different taxonomies, product categories can impact the user behaviour in grid-based product search.

1 Introduction

There is an inherent difference between traditional search engines and e-commerce or product search engines. In traditional search engines like Google, the search results are displayed in a list view whereas in the e-commerce

search engines like www.walmart.com, the results are displayed in a grid view. The difference in the display of search results can impact the customer's attention and behaviour. Researchers in a previous study [?] were able to show that in the Image Search Engine Result Pages (SERPs), the user's attention can be defined by three factors: (1) User's attention within a row is more in the middle, so there is a bias towards the products in the middle (2) Users tend to skip rows while scrolling on a SERP (3) The decay of user attention is slower in a grid view compared to list view, where the decay happens drastically. This idea was further applied in the e-commerce search [?] and developed an inverse propensity bias model to correct the learning to rank algorithm. While position/presentation of the products on a SERP has been considered in the modelling of the bias, the impact of the product's meta-information on the user attention has still not been addressed.

Intuitively, the SERPs of the e-commerce search engines are not only different in the presentation but also in the fact that e-commerce is a marketplace which has a direct monetary benefit for the sellers maintaining the product information on the page. Thus, the information present for each product tile such as 'Top Picks', 'Reviews' and 'Ratings' would have a very strong impact on the customers navigating this web space to get the best product suited for their requirements. Figure 1 and 2 shows the difference in results displayed in list vs grid view on walmart web and application.

The impact of these qualitative product features on bias would be different for different categories of products. For example : On a category of Milk with clear intent, the new customer might not be exploring so much on the view presented whilst on a category of clothing, the user's attention would be influenced by these product tile tags as discussed above. Hence, we extend the framework to model the product meta-information like Ratings, Reviews, Shipping Promise in the propensity model. Incorporating these features into the model will show the impact on the features displayed on the product tile. To model these features, we use a scoring algorithm to assign a score to each product and this score is modelled with the propensity scoring model. Our aim is to learn a ranker and evaluate the impact of the model using an evaluation metric. We also look at the different product taxonomies like Entertainment, Fashion etc because the customer's intent is different in each product taxonomy. We organize the rest of the paper in the following way: Section 2 contains related work on grid based e-commerce search, Section 3 contains our proposed framework with details on how the item feature scores are calculated and how the propensity score for modelling is derived, Section 4 mentions the experimental settings, evaluation metric and the results, along with the dataset description, and Section 5 presents the conclusion and the future work. Each section is further divided into subsections.

2 RELATED WORK

We will start with a brief introduction of the background and knowledge on the prior research work for analyzing the differences in the user behavior patterns in the list view vs the grid view search, and how this behaviour pattern is used in the propensity score modelling with a brief introduction to LambdaMART ranker.

2.1 GRID SEARCH USER BEHAVIOUR PATTERNS

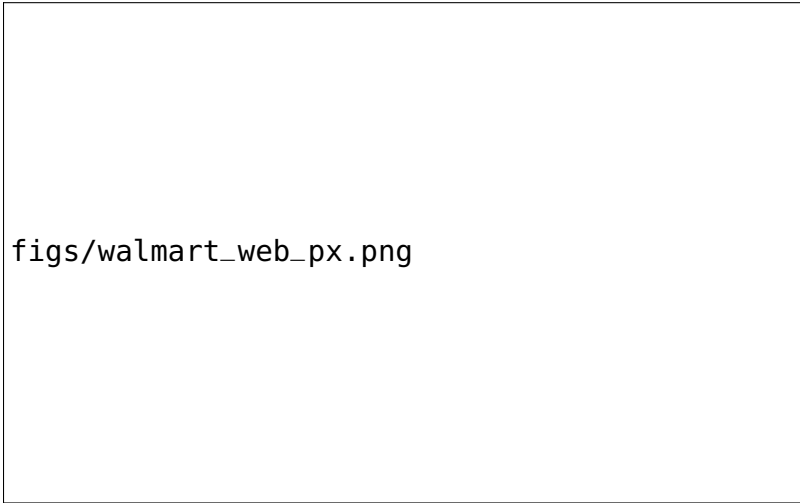
Products in an e-commerce are placed in different view as compared to the majority of search engines that show the list view for the results. The different placement of products leads to a change in the interaction mechanisms of items. Previously, a number of studies have been done on the user behaviour on image or grid search engines [? ? ?]. All these studies focused on comparing the user interaction behaviour in grid search setting to list view display setting by using search logs and eye tracking methods. Important differences in user behaviour, user being more exploratory and more time spent by the user are some of the findings that were observed. We list the major findings in grid based search as follows:

(a) Middle Bias: There exists a "middle position bias" in the user's examination behaviour. Customers allocate their attention more to the items placed in middle positions in a row. The probability of the user examining



figs/walmart_app_px.jpg

Figure 1: List view on Walmart App



figs/walmart_web_px.png

Figure 2: Grid view on Walmart web

items placed on left and right corner in a grid search is less compared to the items in the center. In [?], the researchers consider the dwell time as the factor which is defined as the examination duration of an image and they confirm the middle bias phenomenon. Also, the user behaviour at each position in a row follows normal distribution. Researchers have not used this phenomenon in Learning to rank models of e-commerce yet, and we do not consider it in this research because in the dataset that we use, the number of columns is less than 4 or 5, and our hypothesis is that there are other details in e-commerce search that draw the user attention. So, the middle bias phenomenon is not very effective here.

(b) Row Skipping: The customers do not examine each and every product in a grid from top to bottom, they tend to skip some rows. Users ignore particular rows and directly jump to some distant rows. The probability of the users skipping the first row is very low as compared to the other rows. In [?], row skipping bias in the user behavior was considered in getting an Unbiased Learning to rank model. The probability of the user skipping a

particular row was considered as a bias factor when calculating the lambda gradient for a query. The observation was that the row skipping models performed better in taxonomies where user intent was very clear and users were looking for specific set of items. In such cases, users were probable to skip the rows.

(c) Slower Decay: In a list-view web SERP, the attention of the customer decays in monotonically decreasing fashion and is also dramatic but in grid-view, the decay is much slower. The user behavior in the grid search is more exploratory and there is more browsing as there are a lot of items displayed in a page and users don't have to go to next page to view the next set of items as in the list-view. This leads to the user spending more time in the grid search, concentrating on the items, and thus the decay is slow. In [?], the slower decay user behaviour bias was considered in training unbiased learning to rank models.

These findings are incorporated and Normalized Discounted Cumulative Gain(NDCG) evaluation method is also changed based on grid view search user behaviour.

2.2 Click Models

A lot of prior work has been done on-click modelling in the web search by extracting useful information from search logs of any search engine, mainly the click data. Cascade click model is one of the widely adopted technique in list based search that takes multiple types of user feed backs into account and models the user behavior as a sequence of actions[? ?]. There are different types of biases in the click-models, mainly the position bias. Learning from biased data without taking them into consideration leads to a less effective ranking function. In another study, unbiased learning to rank model was introduced where position bias was considered. Inverse propensity weighting is a well known technique adopted to address any kind of bias and is used for unbiased learning [?]. Most of the work in this area assumes that the propensity scores are present in the logs and they study how to reduce the model variance while remaining unbiased. The unbiased learning-to-rank framework is different, because in that, the propensity is not explicitly logged and is buried in the user behavior data implicitly. Unbiased learning to rank does debiasing of the click data and uses it to train the ranker. Unbiased LambdaMART framework [?] was introduced for training an unbiased ranker by jointly estimating the biases at click positions and the biases at non-click positions.

2.3 LambdaMART Ranker

LambdaMART is a widely used Learning to rank algorithm to solve ranking problems in search engines. LambdaMART [? ? ?] uses gradient boosting and the gradient function of the loss function is called lambda function. The minimization of the objective function is performed using the lambda function on the training dataset. In LambdaMART, the lambda gradient of any document is calculated as

$$\lambda_i = \sum_{j:(d_i, d_j)} \lambda_{ij} - \sum_{j:(d_j, d_i)} \lambda_{ji} \quad (9)$$

λ_{ij} is defined as following :

$$\lambda_{ij} = \frac{-2}{1 + \exp(2(f(x_i) - f(x_j)))} |\Delta_{ij}|$$

where λ_{ij} is the lambda gradient defined on a pair of documents d_i and d_j , σ is a constant, $f(x_i)$ and $f(x_j)$ are the scores of the two documents given by LambdaMART, Δ_{ij} denotes the difference between NDCG scores if documents d_i and d_j are swapped in the ranking list.

3 Item Propensity Scoring Model

We will start with a brief introduction of the background and knowledge on propensity estimation and joint optimization within the learning to rank models. Then we provide description of the changes to the proposed

propensity scoring model using item features.

3.1 Estimating Propensity using Product Meta Information

We hypothesize that the user is not only influenced by the presentation of search query results but also with certain attributes or tags that have been designed to get customer's attention by sellers.

We consider four such attributes about an item which can impact user attention:

(1) **Two Day Shipping:** In the fast paced world that we are in currently, every user wants their favorite products to be delivered at the earliest. Users tend to compare delivery options with other online competitors and place their order accordingly. Thus, to get user attention on items that are eligible for super fast delivery, a message is shown on product tile that this item is eligible for delivery within 2 days. Fig 3,4 shows a two day shipping tag for products in the result set.

(2) **Ratings:** Ratings are given by previous users to the products on any e-commerce platform to indicate how satisfied or dissatisfied previous customers were for that product. This attributes tells what their experience was like and how they rate that experience on a scale of 1 to 5. Fig 3,4 shows the rating tag for products in the result set.

(3) **Reviews:** Reviews is the number of reviews a product has received till date. Reviews is a factor in fetching the rating score for an item. Users are attentive towards number of reviews as well along with rating, as an item with a high rating and very low number of user reviews is highly biased. Products with significant number of reviews gains user trust for that products rating.

(4) **Relative Price:** Price is an important information displayed to the user on the product tile. A set of expensive items for a query may not attract user attention and user may not look in details with much attention. This will counter the slower decay factor. With relative price score, we try to get the expensive and cheap products for a query that is displayed to user. This feature is the relative score among price of all products fetched in the recall set for a particular query.

Based on the above features, we estimate a score and experiment with it in the propensity model. In scoring method, We assign a binary value in the scoring functions for 2-day shipping, ratings and reviews.

According to the formulation:

$$S(I) = \gamma \sum_{i=0}^3 K_i \quad (10)$$

where $K_i \in [-1, 1]$ represents binary vote of the item property.

How to assign the vote for score estimation of an item? To assign a vote for scoring, we look at the quantiles for item features in the data. The threshold for assigning a positive vote or a negative vote is decided using quantile values and then the final score is estimated by the summation of all the votes. The scoring algorithm is mentioned in Algorithm 1.

For example:

If ratings \geq 75th quantile-value, we assign a +1 score to the item or If ratings \leq 25th quantile-value, we assign a -1 score to the item etc. In this similar fashion, we compute a score for each item based on these features.

[H]

- 1: Initialize $score_i$ as 0
- 2: **if** $twoDayShipping_i > 0.0$ **then**
- 3: $score_i \leftarrow score_i + 1$
- 4: **end if**
- 5: **if** $rating_i \geq 75thQuantileValue$ **then**
- 6: $score_i \leftarrow score_i + 1$
- 7: **end if**

figs/4-col-min.png

Figure 3: SERP with 4-column view with product meta information like Ratings, Reviews, Two day shipping

figs/5-col-min.png

Figure 4: SERP with 5-column view with product meta information like Ratings, Reviews, Two day shipping

```

8: if  $rating_i \leq 25thQuantileValue$  then
9:    $score_i \leftarrow score_i - 1$ 
10: end if
11: if  $reviews_i \geq 75thQuantileValue$  then
12:    $score_i \leftarrow score_i + 1$ 
13: end if
14: if  $reviews_i \leq 25thQuantileValue$  then
15:    $score_i \leftarrow score_i - 1$ 
16: end if
17:  $score_i$ 

```

3.2 Changing the Slower Decay Model

As mentioned before, in a SERP with a grid view the user attention decreases more slowly when compared to the results in a list view. According to the earlier work [?], the probability of examination of an item at position, i is estimated as:

$$P(o^i = 1) = \prod_{j=0}^{i-1} \min(\beta^{row(j)} * \alpha), 1.0) \quad (11)$$

where $row(i)$ is the row-number for the item at position i , α is likeness of customer browsing the next item, β models the customer's patience in grid view. As we can see, the probability does not account for item meta-information shown on the item tile in SERP. If an user finds any item attractive in SERP, the user tends to spend more time on it and the probability of examination is more. There are multiple details shown on search page that may attract user attention viz. image quality or attractiveness, user ratings of that particular item, number of reviews on it, special offer tag, price of the product or the delivery date of the product. These factors will also contribute to the probability of user examining any particular product in Grid based SERP. Considering this, the

decay factor will change accordingly. Based on this hypothesis, we model the estimated score above into the probability of examination at position, i as:

$$P(o^i = 1) = \prod_{j=0}^{i-1} \min(\beta^{\text{row}(j)+S(I)} * \alpha), 1.0) \quad (12)$$

We train a ranker f based on loss function L with the propensity score of slower decay model computed using α , β and γ hyper parameters. We use LambdaMART with the ranker, gradient boosting trees as explained before. Following on the same lines, we calculate the lambda gradients by applying inverse propensity scoring. Hence, the lambda gradient for k^{th} product can be re-written as:

$$\frac{\delta L}{\delta x_k} = \lambda_k = \sum_q \left(\sum_{y_q^i = k \cap (i,j) \in I_q} \frac{\lambda_{ij}}{P(o^i)} - \sum_{y_q^j = k \cap (j,i) \in I_q} \frac{\lambda_{ij}}{P(o^j)} \right) \quad (13)$$

where i, j are the positions on the SERP, x_q is the feature vector for query-product pair, y_q^i means that the k^{th} product is at position, i and λ_{ij} is defined as following :

$$\lambda_{ij} = \frac{-2}{1 + \exp(2(f(x_q^i) - f(x_q^j)))} |\Delta_{ij}|$$

Δ_{ij} is the value difference in the NDCG evaluation metric if i and j are to be swapped.

Relative Price Feature : Price is another information displayed on the product tile in an E-commerce search. Most of the users have a budget in mind when they visit to shop any product. If ,for a particular query, most of the items in search results are expensive compared to the range user is looking for, the user may not spend much time on that query viewing the results thus the decay will be comparatively faster. Among the set of items displayed, if there is a relatively cheaper product that also has other attractive tile meta-information, the probability of examining it will be higher. As a part of this work, we consider relative price feature separately to observe the impact of the price on the user behaviour and to check if there is any price bias . Relative price calculation is done as follows:

$$\text{relp}(i) = 1.0 - k * \text{abs}(\text{func}(\log(x/\text{mean_price}))) \quad (14)$$

where $\text{relp}(i)$ is the relative price score for an item i and x is the price of that particular item , k is a constant and mean_price is the mean of prices of all items in the recall set of the query, abs is the absolute function to get the absolute value, func can be any activation function[?] like \tanh or sigmoid . For example , \tanh is defined as

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

Sigmoid is defined as

$$\text{sig}(x) = \frac{1}{1 + e^{-x}}$$

Thus, in our dataset, we have the relative price score for all query-item pairs. Based on this, we redefine the probability of examination at position i in a GRID based SERP as

$$P(o^i = 1) = \prod_{j=0}^{i-1} \min(\beta^{\text{row}(j)+S(I)+\eta*\text{relp}(i)} * \alpha), 1.0) \quad (15)$$

where η is the hyper-parameter introduced to tune the relative price feature in the slower decay model.

For training slower decay model including price feature, we calculate the lambda gradient in the gradient function of LambdaMART as above in (7).

4 Experiments

In this section, we conduct a series of experiments to compare our proposed method with the baseline methods for the removal of bias in grid-based e-commerce search. First, we explain our experimental settings, the datasets that we used, how we prepared them, and the metrics we used to evaluate the models. We then report the results obtained in our unbiased learning to rank setting.

4.1 Datasets

In this subsection, we provide a brief overview of the search log datasets that we used for our experiments. We used the data from the search logs of e-commerce website at Walmart, an American multinational retail corporation with a significant online presence. In particular, we use the data from three product taxonomies viz. Fashion, Entertainment (ENT), and Every Day Living (EDL), and by considering them for capturing and understanding the user behaviour and we eventually obtain four datasets, one for each category. Our datasets included queries, items, positions, and item features. The item features include raw features like item title match, item popularity, partial match score, along with engagement features like click rates, ‘add to cart’ rates, order rates and attribute features like the number of reviews, ratings, the relative price of the products among the recall set, and shipping promise.

4.2 Experimental Settings

In this subsection, we describe the experimental setup. The most common way to evaluate the learning to rank model is by its performance on a holdout set that has unseen data. We perform experiments on each product taxonomy, as mentioned in the subsection above. In our datasets, we perform a random split into a training set (80%), validation set (10%), and test set (10%). We carry out a fast grid search to obtain the best hyper-parameter values for the models. To keep the probability value in a reasonable range, we search alpha in $\{0.8, 0.85, 0.9, 0.95\}$, beta in $\{1.05, 1.1, 1.15, 1.2\}$, and gamma in $\{0.001, 0.003, 0.005, 0.007, 0.01, 0.03, 0.05, 0.07, 0.1\}$. For the LAMBDAMART ranker, we search for the learning rate in $\{0.01, 0.05, 0.1\}$, the maximum depth in $\{3, 5, 7\}$, and the minimum data in leaf in $\{20, 30, 40, 50, 60\}$. We also tune for boosting fraction and feature fraction values in the range 0.7, 0.8, 0.9, 1.0 and 0.8, 0.9, 1.0 respectively. The other parameters are the default settings of Unbiased LambdaMART, while we keep the settings consistent for all the baseline and feature models. We consider the slower decay model[25] as the baseline method. Here, we optimize for the position 10 and consider the metric on the validation set to get the optimal parameter values.

4.3 Evaluation Metrics

Here, we explain the evaluation metric considered for the experiment. We consider the traditional evaluation method of Information Retrieval Systems, Normalized Discounted Cumulative Gain (NDCG) as the evaluation metric [? ? ?]. Experiments are performed in offline settings and we obtain the NDCG metric at different

Taxonomy	α	β	γ
Fashion	0.8	1.05	0.01
EDL	0.95	1.15	0.001
ENT	0.9	1.15	0.03

Table 3: Tuned hyperparameters for item propensity scoring model for different taxonomies

Taxonomy	Framework	NDCG@1	NDCG@3	NDCG@5	NDCG@10
Fashion (w/o price)	Benchmark Slower Decay	0.408	0.486	0.535	0.584
	Proposed framework	0.419	0.492	0.539	0.590
Fashion (with price)	Benchmark Slower Decay	0.408	0.486	0.534	0.583
	Proposed framework(with price)	0.420	0.494	0.540	0.590
Every Day Living	Benchmark Slower Decay	0.553	0.627	0.669	0.711
	Proposed framework	0.553	0.628	0.669	0.711
Entertainment	Benchmark Slower Decay	0.503	0.580	0.624	0.669
	Proposed framework	0.503	0.581	0.624	0.670

Table 4: Experimental Results on different taxonomies using held-out dataset.

positions to compare the proposed framework with the baseline method.

$$NDCG_k = \frac{DCG_k}{IDCG_k} \quad (16)$$

where

$$IDCG_k = \sum_{i=1}^{|REL|} \frac{2^{rel_i} - 1}{\log_2(i + 1)}$$

Here, $IDCG@K$ is the normalizer and $|REL|$ is the list of documents ordered by engagement in the set up to position k . To explain the slower decay of user attention we consider $K = 1, 3, 5, 10$ for the NDCGs.

On the held-out dataset of Walmart Search logs, we calculate the prediction scores of our models. Using the prediction labels, we apply this evaluation method to get the metrics at different positions across different categories.

4.4 Experimental Results

In this section, we provide an overview of the experimental results that we obtain. We share insights on: (1) how e-commerce search results get better with the proposed framework, and (2) how the user behaviour varies across the product taxonomies. We mention the results in Table 2 and summarize our observations as follows:

- Our proposed method for slower decay using product tile information outperforms the baseline method in Fashion vertical by significant margin. This clearly shows the effectiveness of our proposed framework and assumption that user attention decay is slower when user is attracted by any of the item features.
- Performance of our proposed framework in Fashion affirms our initial assumption that users are more inclined on item tile information when shopping for clothing accessories and decay is much slower in such a case as user looks for detailed information.
- The proposed framework of using item attributes in slower decay probability computation shows improvement in Entertainment and Every Day living category too compared to the baseline method of slower decay. Performance improvement is not much compared to Fashion, and this may be because the users have a specific intent when they shop in such categories.
- Adding relative price feature in slower decay probability computation for fashion gave slight improvement over the proposed framework model without price feature. This suggests that price is not a major factor in the users' attempt to click on that product and view the details.

We train separate models for different taxonomies to show that the user behaviour patterns differ across categories and the users react differently to the product tile features.

Hyper Parameter Values: Here, we report the values that we obtain for the hyper parameters (α , β , γ) and that achieve the optimal performance for the proposed framework. For the Fashion category, the proposed slower decay model with $\alpha = 0.8$, $\beta = 1.05$ and $\gamma = 0.01$ outperforms the others. Similarly, for the Every Day Living category, the proposed slower decay model with $\alpha = 0.95$, $\beta = 1.15$ and $\gamma = 0.001$ works best. $\alpha = 0.9$, $\beta = 1.15$ and $\gamma = 0.03$ are the optimal hyper-parameter values for the Entertainment category with the proposed slower decay framework. We experimented on Fashion with the price feature and $\alpha = 0.95$, $\beta = 1.15$, $\gamma = 0.05$, $\eta = 0.01$ are the optimal hyper-parameter values for our dataset.

5 Conclusion and Future work

In this paper, we tried to understand the impact of meta-information on the products displayed on the SERPs for user attention. We extended the solution of debiasing the grid product search by modelling the meta-information of the products, like rating, reviews, shipping promise, price in the propensity model. We showed through extensive experimentation how the different product taxonomies have different user behaviour.

As part of future work, we plan to extend this work to incorporate more product tile features and also use the image quality feature to model the user attention in Grid-based E-commerce search engines. We plan to try out different scoring methods as well for tile features score calculation and model middle bias, row skipping behaviour based on product tile features.

References

- [1] K. Järvelin and J. Kekäläinen. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.*, 10.1145/582415.582418, 2002.
- [2] C. JC Burges. From RankNet to LambdaRank to LambdaMART: An Overview. <https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambda-rank-to-lambda-mart-an-overview/>, MSR-TR-2010-82, 2010.
- [3] Z. Cao, T. Qin, T.Y. Liu, M.F. Tsai and H. Li. Learning to Rank: From Pairwise Approach to Listwise Approach. *Proceedings of the 24th International Conference on Machine Learning*, 10.1145/1273496.1273513, 2007.
- [4] Z. Hu, Y. Wang, Q. Peng and H. Li. Unbiased LambdaMART: An Unbiased Pairwise Learning-to-Rank Algorithm. *The World Wide Web Conference*, 10.1145/3308558.3313447, 2019.
- [5] X. Xie, J. Mao, Y. Liu, M. de Rijke, Y. Shao, Z.-Ye, M. Zhang, S. Ma. Grid-Based Evaluation Metrics for Web Image Search. *The World Wide Web Conference*, 10.1145/3308558.3313514, 2019.
- [6] R. Guo, X. Zhao, A. Henderson, L. Hong and H. Liu. Debiasing Grid-Based Product Search in E-Commerce. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 10.1145/3394486.3403336, 2020.
- [7] X. Wang, N. Golbandi, M. Bendersky, D. Metzler and M.-Najork. Position Bias Estimation for Unbiased Learning to Rank in Personal Search. *Association for Computing Machinery*, 10.1145/3159652.3159732, 2018.
- [8] B. Geng, L. Yang, C. Xu, X.S. Hua and S.-Li. The Role of Attractiveness in Web Image Search. *Proceedings of the 19th ACM International Conference on Multimedia*, 10.1145/2072298.2072308, 2011.
- [9] X. Xie, J. Mao, Y. Liu, M. de Rijke, H. Chen, M. Zhang, S. Ma. Preference-Based Evaluation Metrics for Web Image Search. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 10.1145/3397271.3401146, 2020.
- [10] N. Craswell, O. Zoeter, M. Taylor and B. Ramsey. An Experimental Comparison of Click Position-Bias Models. *Proceedings of the 2008 International Conference on Web Search and Data Mining*, 10.1145/1341531.1341545, 2008.
- [11] S. K. K. Santu, P. Sondhi and C. Zhai. On Application of Learning to Rank for E-Commerce Search. *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 10.1145/3077136.3080838, 2017.

- [12] Sham. S. Discounted Cumulative Gain. <https://machinelearningmedium.com/2017/07/24/discounted-cumulative-gain/>, 2017.
- [13] Chandekar. Pranay. Evaluate your Recommendation Engine using NDCG. [tps://towardsdatascience.com/evaluate-your-recommendation-engine-using-ndcg-759a851452d1](https://towardsdatascience.com/evaluate-your-recommendation-engine-using-ndcg-759a851452d1), 2020.
- [14] A. Sharma V. Understanding Activation Functions in Neural Networks. <https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0>, 2017.

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

Xu Liu¹, Congzhe Su², Amey Barapatre², Xiaoting Zhao², Diane Hu², Chu-Cheng Hsieh², Jingrui He³

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 [? ? ? ? ?].

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 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 “wristwate” must produce



figs/fig_ranking_system2.pdf

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.

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 ??): 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 [? ? ?]; 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) [? ? ? ?]. 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 *Online Personalized Attribute-based Re-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 [?], product category [? ?] or a context [? ? ?]. 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 ?? 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 ??). 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 [? ?]. Significant

breakthroughs in deep learning (i.e, batch normalization and dropout), have led to its wide adoptions in various communities and applications [?]. In [?], 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 [? ? ? ? ?].

Given that a long-term memory models are insufficient to address drift in user interests, [?] 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, [?] 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, [?] 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, [?] 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 [?] proposed a Matrix Factorization-based attention model to address large-volume and high-velocity session streaming data and [?] 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 [?] 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 [?], Upper Confidence Bounds [?], Thompson sampling [?], EXP3 [?], and others [?]. In the e-commerce [?] 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 [?] 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 [?] 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 [?] 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. [?] proposed an interactive collaborative topic regression model that infers the clusters of arms via topic models [?] and then utilizes dependent arms for the recommendation.

In this literature, it is common to address this problem by treating each arm in the bandit to represent a single item [?], product category [?] or a context [? ? ?]. 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

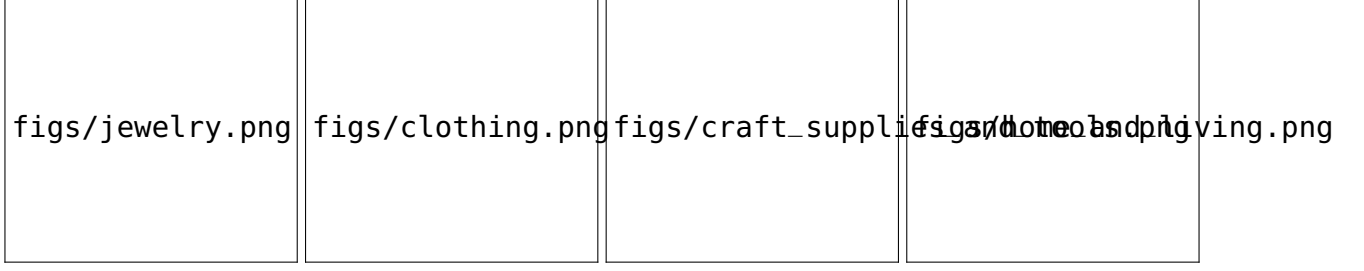


Figure 3: Top attribute-value pairs for top categories: (1) Jewelry; (2) Clothing; (3) Craft Supplies and Tools, (4) Home and Living.

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 :

$$A_t(x_i) = \begin{cases} 0, & \text{no action on } x_i \\ 1, & x_i \text{ is purchased} \\ 2, & x_i \text{ is added to cart} \\ 3, & x_i \text{ is clicked} \end{cases}, \forall x_i \in I_t. \quad (17)$$

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 : $x_i = \{atr_1, atr_2, \dots, atr_{H_{x_i}}\}$.

Figure ?? 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 Θ on attributes as beta-distributed: $\Theta = \{\theta_{atr_n}\}_{n=1}^N \sim \{Beta(\alpha_{atr_n}, \beta_{atr_n})\}_{n=1}^N$, 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 [?], let θ_{atr_n} be beta-distributed, with $\alpha_{atr_n}, \beta_{atr_n}$ be the two parameters of the distribution. In Section ?? 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 Θ . 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 Θ , 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 Θ to optimize in-session personalization?

Input: At time t , (1) Candidate list of items I_t , and (2) user in-session preference Θ .

Output: Sequentially learn $f_t : I_t \times \Theta \rightarrow \tilde{I}_t$.

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.

[!t]

Given a session $S = \{[Q_t, I_t, A_t]\}_{t=1}^T$

$\{\delta_i\}_{i: \text{actions}}$: action-aware increments on attribute parameters

γ : 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

$[Q_t, I_t, A_t] \in S$ **(1) Rerank on the Item List** $f : I_t \rightarrow \tilde{I}_t$

sample $s_{atr_h} \sim Beta(\alpha_{atr_h}, \beta_{atr_h}), \forall atr_h \in N_S$

$x_i \in I_t$ Given $x_i = \{atr_h\}_{h=1}^{H_{x_i}}$ as associated attributes in x_i , set $score(x_i) = \sum_{atr_h \in x_i} g(s_{atr_h})$ $\tilde{I}_t = \text{sorted}([score(x_i)]_{x_i \in I_t})$

(2) Update attribute parameters given A_t

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

Let $\mathcal{V}_t = \cup\{atr_h : \forall atr_h \in x_i \forall x_i \in I_t\}$

$x_i \in I_t$ $A_t(x_i) \neq 0$, item x_i has positive actions $\alpha_{atr_h} + = \delta_{A_t(x_i)} \times \{1 - \text{Exp}(-|\mathcal{U}_t|_0)\}, \forall atr_h \in x_i$

$A_t(x_i) = 0$, no action on item x_i $\beta_{atr_h} + = \delta_{A_t(x_i)} \times \{1 - \text{Exp}(-\gamma|\mathcal{V}_t \setminus \mathcal{U}_t|_0)\}, \forall atr_h \in x_i$ **end** All

re-ranking results $[\tilde{I}_t]_{t=1}^T$

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 [?].

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 :

$$\theta_{atr_h} \sim Beta(\alpha_{atr_h}, \beta_{atr_h}). \quad (18)$$

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

$$score(x_i) = \sum_{atr_h \in x_i} g(\theta_{atr_h}), \quad (19)$$

where $g(\theta_{atr_h}) = \frac{1}{rank(\theta_{atr_h})}$ is a harmonic 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 satisfaction 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 $[score(x_i)]_{x_i \in I_t}$.

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 $x_i \in I_t$:

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

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,

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

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

$$\begin{aligned} \alpha_{atr_h} + &= \sum_{\tilde{\mathcal{Y}}_{t,atr_h}} \delta_{A_t(x_i)} \left(1 - e^{-|\mathcal{U}_t|_0}\right), \forall atr_h \in \mathcal{U}_t, \\ \beta_{atr_h} + &= \sum_{\tilde{\mathcal{Z}}_{t,atr_h}} \delta_{A_t(x_i)} \left(1 - e^{-\gamma|\mathcal{V}_t \setminus \mathcal{U}_t|_0}\right), \forall atr_h \in \mathcal{V}_t \setminus \mathcal{U}_t, \end{aligned} \quad (20)$$

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

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

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.

Table 5: 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

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 ??*)

Experiment #2: How does *OPAR* perform as an action-aware model? (*Answered in subsection ??*)

Experiment #3: How does *OPAR* help to understand users’ short-term, in-session shopping preference? (*Answered in subsection ??*)

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 ?? 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 ?. The remaining data is the **testing dataset**, on which we report re-ranking performance for *OPAR* and other baseline algorithms on in Table ??.

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 ??). 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) [?], 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:

$$\text{click-NDCG}_t = \text{click-DCG}_t / \text{IDCG}_t, \forall t = 1, \dots, T, \quad (21)$$

and *click-NDCG* of a session S is the average of *click-NDCG_t* over events that have at least one click:

$$\text{click-NDCG} = \text{Average}(\text{click-NDCG}_t). \quad (22)$$

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 ??, 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 [?] is the boosted tree version of LambdaRank [?], 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.
2. Atr-KNN is derived from Item-KNN [?]. 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.

Table 6: Re-ranking performance comparison on over all data sets (top-left) and 7 category-specific data sets.

		Over All Category						Clothing					
		LambdaMART	Atr-KNN	Atr-POP	GRU4Rec	OPAR	OPAR _w	LambdaMART	Atr-KNN	Atr-POP	GRU4Rec	OPAR	OPAR _w
4* Purchase NDCG	@4	0.1795	0.0130	0.0749	0.0618	0.2994	0.3042	0.1948	0.0103	0.0516	0.0551	0.2384	0.2494
	2-15 @12	0.2629	0.0412	0.1323	0.1425	0.3505	0.3607	0.2670	0.0348	0.1269	0.0824	0.2685	0.2744
	2-15 @24	0.3162	0.1260	0.2112	0.2018	0.3718	0.3900	0.3019	0.0090	0.2193	0.1434	0.3209	0.3263
	2-15 @48	0.3724	0.2554	0.2861	0.2518	0.4512	0.4578	0.3774	0.2462	0.2784	0.2157	0.3976	0.4030
4* Click NDCG	@4	0.1459	0.0816	0.0705	0.0701	0.3120	0.3158	0.1328	0.0067	0.0690	0.0691	0.3058	0.3197
	2-15 @12	0.2265	0.1456	0.1264	0.1354	0.3213	0.3229	0.2137	0.0228	0.1224	0.1414	0.3126	0.3257
	2-15 @24	0.2955	0.2157	0.2021	0.1922	0.3318	0.3489	0.2821	0.0658	0.2045	0.1844	0.3274	0.3424
	2-15 @48	0.3815	0.3245	0.2813	0.2689	0.4047	0.4051	0.3711	0.2309	0.2807	0.2613	0.3988	0.4061
		Home & Living						Paper & Party Supplies					
		LambdaMART	Atr-KNN	Atr-POP	GRU4Rec	OPAR	OPAR _w	LambdaMART	Atr-KNN	Atr-POP	GRU4Rec	OPAR	OPAR _w
4* Purchase NDCG	@4	0.1755	0.0131	0.0649	0.0571	0.2920	0.2952	0.1822	0.0010	0.1255	0.0684	0.2828	0.2965
	2-15 @12	0.2670	0.0396	0.1226	0.1281	0.3391	0.3436	0.2667	0.0406	0.1692	0.0941	0.3367	0.3497
	2-15 @24	0.3218	0.0936	0.2066	0.1752	0.3838	0.3879	0.3276	0.1297	0.2469	0.1542	0.3796	0.3905
	2-15 @48	0.3874	0.2543	0.2789	0.2164	0.4462	0.4491	0.3876	0.2550	0.3216	0.1943	0.4291	0.4399
4* Click NDCG	@4	0.1481	0.0054	0.0601	0.0944	0.3201	0.3219	0.1585	0.0052	0.1084	0.0839	0.2825	0.2874
	2-15 @12	0.2294	0.0213	0.1175	0.1416	0.3244	0.3256	0.2394	0.0247	0.1586	0.1367	0.2931	0.2973
	2-15 @24	0.2978	0.0598	0.1973	0.1843	0.3485	0.3491	0.3103	0.0644	0.2300	0.1742	0.3383	0.3189
	2-15 @48	0.3835	0.2278	0.2746	0.2288	0.4032	0.4086	0.3911	0.2306	0.3104	0.2007	0.4017	0.4072
		Craft Supplies & Tools						Accessories					
		LambdaMART	Atr-KNN	Atr-POP	GRU4Rec	OPAR	OPAR _w	LambdaMART	Atr-KNN	Atr-POP	GRU4Rec	OPAR	OPAR _w
4* Purchase NDCG	@4	0.1912	0.0135	0.0739	0.0741	0.3101	0.3268	0.1954	0.0251	0.0683	0.0511	0.2166	0.2178
	2-15 @12	0.2735	0.0407	0.1296	0.1125	0.3673	0.3781	0.2828	0.0741	0.1431	0.0849	0.2835	0.2930
	2-15 @24	0.3272	0.1208	0.1970	0.1644	0.4084	0.4188	0.3324	0.1406	0.2510	0.1222	0.3304	0.3361
	2-15 @48	0.3844	0.2577	0.2820	0.2214	0.4366	0.4750	0.3869	0.2693	0.2917	0.1641	0.3962	0.4020
4* Click NDCG	@4	0.1458	0.0055	0.0749	0.0994	0.3118	0.3166	0.1502	0.0105	0.0673	0.0712	0.2495	0.2605
	2-15 @12	0.2262	0.0513	0.1290	0.1279	0.3241	0.3293	0.2324	0.0439	0.1358	0.1331	0.2656	0.2708
	2-15 @24	0.2955	0.2042	0.1953	0.1935	0.3525	0.3521	0.3006	0.1091	0.2398	0.1800	0.3155	0.3212
	2-15 @48	0.3811	0.2278	0.2815	0.2277	0.4080	0.4078	0.3848	0.2391	0.2885	0.2312	0.3548	0.4029
		Electronics & Accessories						Jewelry					
		LambdaMART	Atr-KNN	Atr-POP	GRU4Rec	OPAR	OPAR _w	LambdaMART	Atr-KNN	Atr-POP	GRU4Rec	OPAR	OPAR _w
4* Purchase NDCG	@4	0.2136	0.0501	0.0715	0.0814	0.2847	0.2995	0.1661	0.0060	0.0576	0.0718	0.3051	0.3285
	2-15 @12	0.3014	0.1109	0.1546	0.1223	0.3386	0.3782	0.2534	0.0766	0.1074	0.1142	0.3484	0.3854
	2-15 @24	0.3519	0.0176	0.2652	0.1674	0.4257	0.4152	0.3087	0.1470	0.1668	0.1847	0.3866	0.3973
	2-15 @48	0.4060	0.2965	0.2981	0.2416	0.4516	0.4656	0.3814	0.2460	0.2663	0.2367	0.4425	0.4598
4* Click NDCG	@4	0.1530	0.0267	0.0805	0.0641	0.2074	0.2051	0.0701	0.0027	0.0621	0.0614	0.3314	0.3892
	2-15 @12	0.2324	0.0703	0.1580	0.0939	0.2487	0.2622	0.1314	0.0106	0.1141	0.1021	0.3783	0.3963
	2-15 @24	0.3029	0.1410	0.2657	0.1345	0.3158	0.3120	0.1989	0.1276	0.1762	0.1647	0.3956	0.4162
	2-15 @48	0.3880	0.2560	0.3026	0.1667	0.3978	0.4078	0.3119	0.2192	0.2700	0.2144	0.4190	0.4475

- 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 [?] given its simplicity and efficacy.
- GRU4Rec [?] 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.

Table 7: Multiple Purchase Intents within One Session

	Timestamp	Query	Query Taxonomy	Engaged Attributes
2-6 4* 1st Purchase Intent	0	'flower girl basket'	paper and party supplies (NO ACTION)	Browsing
2-6	1-4	'flower girl basket wedding'	paper and party supplies (CLICK)	'Prime Color: White', 'Occasion: Wedding', 'Holiday: Christmas', 'Wedding theme: Beach & tropical', 'Craft type: Floral arranging'
2-6	5-9	'flower girl basket beach wedding'	paper and party supplies (CLICK)	'Prime Color: Blue', 'Occasion: Wedding', 'Holiday: Christmas', 'Wedding theme: Beach & tropical', 'Secondary color: White', 'Craft type: Floral arranging'
2-6	10-11	'two flower girl and one pillow'	paper and party supplies	Browsing
2-6	Purchase Intent Change			
2-6 3* 2nd Purchase Intent	12-15	'hat for beach wedding'	clothing.women_clothing (CLICK)	'Prime Color: Blue', 'Occasion: Wedding'
2-6	16-22	'turquoise petals'	accessories (CLICK)	'Prime Color: Blue', 'occasion: Bridal shower', 'Wedding theme: Fairytale & princess'
2-6	23	'bride hair decoration beach theme'	clothing.women_clothing (NO ACTION)	Browsing
2-6 2* Final Purchase	24	'turquoise petals'	accessories (PURCHASE)	'Prime Color: Blue', 'Occasion: Bridal shower', 'Wedding theme: Fairytale & princess'

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 ?? shows experiment results of our model (*OPARs*) against 4 baselines described in Section ?. 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 achieved 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.

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 ?? 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 ??, 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 ?? presents a record of a user's in-session activities. Figure ?? (top) shows the sequential improvement of *OPAR* in session-level click-NDCG over time compared to the baseline, and Figure ?? (bottom) shows how *OPAR*

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

captures user’s preference, θ_{atr_h} , on 5 attributes over time. The “Engaged Attributes” column in Table ?? maps out all attributes associated with the clicked items for the corresponding query.

As shown in Table ??, 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 $t = 5, \dots, 9$ 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 ?? (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 & 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 & 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. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & 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 & Data Mining, 2019.
- [22] P. Pobrotyn, T. Bartczak, M. Synowiec, R. Białobrzski, 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 & Data Mining*, 2020.

shortcuts,acronym]glossaries

WDCWDCWeb Data Commons MWPDMWPDMining the Web of Product Data GPCGPCGS1 Global Product Classification standard GTINGTINGlobal Trade Item Number MLMMLMMasked Language Modelling BPEBPEByte-Pair Encoding RNNRNNRecurrent Neural Network NLPNLPNatural Language Processing wF1wF1average weighted F1 hF1hF1hierarchical F1

Copyright 2021 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Improving Hierarchical Product Classification using Domain-specific Language Modelling

Alexander Brinkmann, Christian Bizer
University of Mannheim
{alex.brinkmann, chris}@informatik.uni-mannheim.de

Abstract

In order to deliver a coherent user experience, product aggregators such as market places or price portals integrate product offers from many web shops into a single product categorization hierarchy. Recently, transformer models have shown remarkable performance on various NLP tasks. These models are pre-trained on huge cross-domain text corpora using self-supervised learning and fine-tuned afterwards for specific downstream tasks. Research from other application domains indicates that additional self-supervised pre-training using domain-specific text corpora can further increase downstream performance without requiring additional task-specific training data. In this paper, we first show that transformers outperform a more traditional fastText-based classification technique on the task of assigning product offers from different web shops into a product hierarchy. Afterwards, we investigate whether it is possible to improve the performance of the transformer models by performing additional self-supervised pre-training using different corpora of product offers, which were extracted from the Common Crawl. Our experiments show that by using large numbers of related product offers for masked language modelling, it is possible to increase the performance of the transformer models by 1.22% in wF1 and 1.36% in hF1 reaching a performance of nearly 89% wF1.

1 Introduction

Product aggregators like market places or price portals support customers in finding the right offer for their desired product. To ensure a good customer experience, product aggregators integrate heterogeneous product offers from large numbers of online shops into their own product categorization hierarchy. This hierarchical product classification task is a major challenge for product aggregators as most shops use their own proprietary categorization hierarchy as well as diverse titles and descriptions for the same product. A promising technique to improve hierarchical product classification are pre-trained transformer models [? ?]. These pre-trained transformer models have recently shown success for many NLP tasks [? ? ? ? ?]. The training of transformer models involves two steps [? ?]:

1. Pre-Training: The model is pre-trained on a huge corpus of texts from books, news, online forums and stories using self-supervised MLM.
2. Fine-Tuning: The pre-trained model is fine-tuned for downstream tasks using task-specific training data.

During pre-training the model acquires general knowledge on language representation. This knowledge can be applied to solve down-stream tasks. In related work the pre-training step is extended by additionally pre-training the transformer model on domain-specific text corpora [? ? ? ?]. In these works, the extended self-supervised pre-training results in improved performance on downstream tasks.

Motivated by these findings, we investigate whether additional pre-training using heterogeneous product offers from the Web can improve hierarchical product classification. For this purpose, we use product offers, which the Web Data Commons project¹ has extracted from the Common Crawl². For identifying product offers and their attributes, the project relies on schema.org³ annotations in the HTML pages of the web shops [?]. The annotations enable the reliable extraction of the offer’s title, the description of the offered product, as well as the offer’s categorization within the proprietary categorization hierarchy of the specific web shop. The heterogeneous category values are of special interest, because the categories contain information about the product classification of the web shop. While being heterogeneous and web shop specific, previous work has show that this knowledge about product categories is beneficial for categorizing products into a single central product hierarchy [? ?].

We experiment with three different product corpora for pre-training that differ in size and relatedness to the downstream task. Through these different characteristics we measure the influence of size and relatedness of the pre-training corpus on the downstream hierarchical product classification task. Additionally, we experiment with different hierarchical classification methods. The methods combine RoBERTa_{base} [?] with various classification heads in order to evaluate different approaches for exploiting the product hierarchy. We evaluate the classification methods using two product classification tasks involving product offers from many different web shops.

The contributions of this paper are as follows:

- We are the first to show that the performance of transformer models can be improved for the task of hierarchical product classification by performing additional pre-training using a corpus of related product offers.
- We show that using related product offers results in a better performance compared to randomly sampled product offers.

This paper is structured as follows: Section ?? introduces the classification models that will later be used for the experiments. Section ?? describes the evaluation tasks. While Section ?? presents the results of baseline experiments without additional language modelling. The effects of domain-specific MLM for hierarchical product classification are investigated in Section ?. Section ?? discusses related work. All data and code needed to replicate the results are available online⁴.

2 Classification models

The architecture of all classification models is composed of a pre-trained RoBERTa_{base} transformer model and a task-specific classification head. RoBERTa_{base} is chosen due to its recent success on related NLP tasks [?]. Additionally, for one baseline model RoBERTa_{base} is replaced by fastText⁵, a state of the art neural network architecture for language representations [?]. For the classification the RoBERTa_{base} model encodes the input text of the product offer. The first token of the encoded product offer is handed over to the classification head. Based on the first token, also referred to as [CLS] token, the classification head predicts a category for each product hierarchy level. Since it can be assumed that the product hierarchy contains valuable information, the given product classification challenge is tackled with three different classification heads. These classification heads try to exploit the upfront known product hierarchy. The three approaches are referred to as flat classification, hierarchical softmax and RNN.

¹<http://webdatacommons.org/largescaleproductcorpus/v2/>

²<https://commoncrawl.org/>

³<https://schema.org/>

⁴<https://github.com/wbsg-uni-mannheim/productCategorization>

⁵<https://github.com/facebookresearch/fastText>

2.1 Flat Classification

For the flat classification, a linear layer makes a prediction based on the [CLS] token. As this classification approach only assigns product offers to lowest level of categories, the parent categories are inferred using the product hierarchy. For training a cross-entropy loss is used.

2.2 Hierarchical Softmax

The hierarchical softmax classification head predicts the category path with the highest probability for a product offer. To arrive at a probability for a category path, one local classifier is trained for each category in the product hierarchy. The local classifier predicts the probability of a category to be part of the category path. The product of all local predictions along a category path is the probability of a category path to be predicted for a product offer. Using softmax the most probable category path among all category paths is chosen. The input of the local classifiers is the transformer's [CLS] token. For training the cross-entropy loss is calculated per local classifier and per global category path. The combined loss deals with both the local impact of a single classifier and the global impact of a combination of classifiers along a category path.

2.3 Recurrent Neural Network

For the third classification head a RNN sequentially predicts a category for each level in the product hierarchy. The input for this classification head are the transformer's [CLS] token and a hidden state with the same size as the [CLS] token. Based [CLS] token and hidden state a linear layer predicts the category for the current product hierarchy level. A second linear layer updates the hidden state. The updated hidden state is fed back into the RNN to predict the next level in the product hierarchy. This procedure is repeated until a category is predicted for each level in the product hierarchy. During training the cross-entropy loss is calculated for each predicted category.

3 Evaluation Tasks

This section introduces the hierarchical product classification tasks that are used for the evaluation. The objective of the tasks is to assign product offers from different web shops to the correct categories in a single central product hierarchy.

3.1 MWPD Task

The MWPD task was used at the MWPD challenge⁶ [?] for benchmarking. The MWPD challenge was part of the International Semantic Web Conference (ISWC2020). In the product classification task of the MWPD challenge participants have to sort product offers from different web shops into the GPC⁷ [?]. GPC classifies product offers into a product hierarchy based on their essential properties and their relationship to other products. For the gold standard of the MWPD product classification data set the extracted product offers are manually assigned to the first three levels of the GPC.

3.2 Icecat/WDC222 Task

The training set of the Icecat/WDC222 task⁸ is built based on the Open Icecat product data catalogue⁹. The Open Icecat product data catalog provides well maintained and normalized product information. For this work

⁶<https://ir-ischool-uos.github.io/mwpd/>

⁷<https://www.gsl.org/standards/gpc>

⁸<http://data.dws.informatik.uni-mannheim.de/largescaleproductcorpus/categorization/>

⁹<https://icecat.biz/en/menu/channelpartners/index.html>

the attributes title, description, category and GTIN are considered. For the classification the first three levels of the product hierarchy are considered. In order to evaluate whether a classifier is able to correctly classify heterogeneous product offers, the test set of the Icecat/WDC222 task consists of selected product offers from the WDC Product Corpus¹⁰. This corpus contains offers from 79 thousand different websites, which use schema.org annotations. Using the GTIN the product offers are assigned to one out of 222 leaf categories in the Icecat product hierarchy. All assignments are manually verified. For all product offers the values of the attributes title, descriptions and GTIN are extracted. As the Icecat training set contains normalized product offers and the WDC222 test set contains heterogeneous product offers, the Icecat/WDC222 task measures the transferability of a classifier trained on clean product offers and transferred to a scenario involving heterogeneous product offers.

Table ?? shows that the MWPD training set is small compared to the training set of the Icecat use case, but the product offers of the MWPD task are drawn from a comparably large number of different hosts. The WDC222 test set is again rather small but covers more hosts than the Icecat training set. These high numbers of hosts are an indication for more heterogeneity, because the product offers are differently represented by different hosts. The analysis of the median and maximum number of records per category of both use cases as shown in Table ?? reveals that the distribution of product offers among the categories is skewed towards a small number of categories. This distribution is common for hierarchical classification tasks [?]. The missing description values of the Icecat/WDC222 task are a sign that the description might harm a classifier’s performance if the classifier is trained on the Icecat training set and applied to the WDC222 test set.

Evaluation Task	No. Records Train Set	No. Records Test Set	No. Hosts Train Set	No. Hosts Test Set	No. Nodes in Hierarchy	Avg. Depth Hierarchy
MWPD	10,012	3,107	1,547	878	396	3
Icecat/WDC222	765,743	2,984	1	112	410	2.44

Table 8: Evaluation Task Statistics

Evaluation Task	Data Set	Median No. Characters Title	Missing Values Description	Median No. Characters Description	Median No. Records per Category	Maximum No. Records per Category
MWPD	Train	50	0%	304	7	3,228
MWPD	Test	48	0%	365	4	799
Icecat/WDC222	Train	57	29.65%	1,099	215	145,020
Icecat/WDC222	Test	54	22.72%	140.5	3	516

Table 9: Attribute Statistics

4 Baseline Experiments

In order to set baselines, we apply the classification models described in Section ?? to both evaluation tasks that were introduced in Section ?. This section describes the setup as well as the results of the baseline experiments.

4.1 Evaluation Metrics

We use the wF1 score and the hF1 score to evaluate the performance of the different models. Both scores are designed for hierarchical classification tasks [? ?]. The wF1 score is calculated as proposed by the organisers of

¹⁰<http://webdatacommons.org/largescaleproductcorpus/v2/>

the MWPD challenge and shown in equation ?? [?].

$$\text{Average weighted F1 (wF1)} = \sum_{j=1}^L \frac{1}{L} \sum_{i=1}^{K_j} \frac{n_i}{N} F_i \quad (23)$$

First, the F1 score of every category i in the hierarchy is calculated. To calculate the weighted F1 score per hierarchy level, the $F1_i$ score for each category i is weighted by number of true instances n_i for each category i divided by the total number of instances N across all categories K on a specific hierarchy level. For the hF1 score all target and prediction categories of the different levels in the product hierarchy are considered to calculate the F1 score. This way the hF1 score is suitable for hierarchical classification tasks, as it directs higher credit to partially correct classifications, considers the distance of errors to the correct category and errors higher up in the hierarchy are punished more severely [?]. Additionally, McNemar’s significance test is applied to verify significantly different model performances on the test set [?]. For the test it is determined if a classifier’s prediction is correct or incorrect first. Second, the numbers of correctly predicted product offers by the first classifier and incorrectly predicted product offers by the second classifier (correct/ incorrect) and vice versa (incorrect/ correct) are calculated. Using these numbers of correct/ incorrect and incorrect/ correct predictions as well as a significance level of 0.01, McNemar’s test determines if the proportion of errors and consequently the performance of the two compared classifiers on the test set is significantly different.

4.2 Experimental Setup

We use the following hyperparameter setting for the experiments: The learning rate is set to $3e-5$ for the Icecat/WDC222 task and to $5e-5$ for the MWPD task. We use a batch size of 8 and a linear weight decay of 0.01. All fine-tuning experiments are run for 25 epochs on the MWPD data set and 10 epochs on the Icecat/WDC222 data set. The different learning rates and numbers of epochs are a result of multiple experiment runs. In this setting the average results on the test set over three randomly initialized runs are reported for every experiment. For McNemar’s test a majority voting among the results of the different runs is performed. As input for the classification models the values of the attributes title and description are lowercased and excessive white-spaces are removed. For the experiments in this section a RoBERTa_{base} model is used to obtain a product representation, which is consumed by different classification heads to obtain a classification.

4.3 Results

The naming convention <input attributes>-<transformer model>-<head> is used to refer to the different models. If the value of <input attribute> is "1", only the title is used as input. If the value of <input attribute> is "2", both title and description are used as input. In this section <transformer model> is either "base" for RoBERTa_{base} or "fast" for fastText. The value of <head> refers to one of the classification heads introduced in Section ?? "flat", "hier" for hierarchical or "rnn". Experiments with the same capital letter in the column "Same Error Rate" share the same error proportion on the test set according to the significance test. Otherwise the experiment’s error proportion is significantly different. The experimental results for the MWPD task are shown in Table ??. Setting the results of the model 2-fast-flat into context to the other models shows that all transformer-based approaches outperform the fastText baseline model. A comparison of the models 1-base-flat and 2-base-flat reveals that adding the description as input improves the performance of the classification. The performance difference of the models 2-base-flat and 2-base-rnn is not significant and 2-base-hierarchical performs worse than the other two models. Thus, 2-base-flat is chosen as baseline model for the experiments with domain-specific language modelling as described in Section ??. Table ?? shows the results of the different models for the Icecat/WDC222 task. Again, the fastText based model 1-fast-flat is outperformed by the transformer-based models. The comparison of the models 1-base-flat and 2-base-flat shows that adding the description harms the

performance of the trained classifier. This finding is expected given the high percentage of missing values and the difference in the median number of characters between training and test set as shown in Table ?? . This comparison of the models 1-base-flat and 1-base-rnn shows that the RNN leads to a performance gain. A reason for this improvement might be the huge size of the Icecat training data set compared to the size of the MWPD data set. This size enables the RNN classification head to better learn the encoded hierarchy of the labels, which is beneficial for the classification on the test data set.

Model	Attributes	Classification Head	wF1	Δ wF1	hF1	Δ hF1	Same Error Rate
2-fast-flat	Title, Desc.	Flat	84.26		82.68		
1-base-flat	Title	Flat	87.01	2.75	87.03	4.35	
2-base-flat	Title, Desc.	Flat	87.52	3.26	87.62	4.94	A
2-base-hier	Title, Desc.	Hierarchical	87.00	2.74	87.47	4.79	
2-base-rnn	Title, Desc.	RNN	87.47	3.21	87.67	4.99	A

Table 10: Experimental results without Language Modelling - MWPD Task

Model	Attributes	Classification Head	wF1	Δ wF1	hF1	Δ hF1	Same Error Rate
1-fast-flat	Title	Flat	77.58		83.64		
1-base-flat	Title	Flat	83.36	5.78	84.69	1.05	
2-base-flat	Title, Desc.	Flat	80.91	3.33	81.48	-2.16	
1-base-rnn	Title	RNN	86.56	8.98	85.61	1.97	

Table 11: Experimental results without Language Modelling - Icecat/WDC222 Task

5 Domain-specific Language Modelling

After establishing baseline results in the previous section, we now investigate the effect of domain-specific language modelling on the performance of RoBERTa_{base} models for hierarchical product classification. In this section the extraction of the domain-specific product offer corpora and the applied MLM approach are explained. The effects of domain-specific MLM are demonstrated by fine-tuning the newly pre-trained transformer models on the MWPD use case. The results of this fine-tuning are set into relation to the baseline results without domain-specific pre-training.

5.1 Product Corpora

In total three different product corpora are used for domain-specific MLM. All three product corpora contain product offers extracted from the WDC Product Corpus¹¹. The WDC Product Corpus contains structured data for 365,577,281 product offers that are extracted from 581,482 different hosts using schema.org annotations. The hosts use schema.org annotations to enrich the search results of product aggregators with their web shop’s product offers [?]. Three strategies are applied to retrieve product offers from the WDC Product Corpus. For the first two domain-specific product corpora 1,547 top-level domains of product offers from the MWPD training set are identified. Using these top-level domains, product offers from the same top-level domains are extracted from the WDC Product Corpus. This heuristic assumes that all products offered by a single shop (top-level

¹¹http://webdatacommons.org/structureddata/2017-12/stats/schema_org_subsets.html

domain) are related. The first product corpus contains 75,248 product offers and is called Small Related product corpus. The second product corpus contains 1,185,884 product offers and is referred to as Large Related product corpus. Through these two corpora the effect of the number of the product offer on MLM is measured. For the third corpus a large random sample of product offers is extracted from the WDC product corpus. This corpus is referred to as Large Random product corpus. The Large Random product corpus enables us to measure the effect of relatedness of product offers on domain-specific language modelling. Table ?? gives an overview of the product corpora’s characteristics. For the two related product corpora the median number of records per hosts is higher compared to the Large Random product corpus. This shows the focus of the related corpora on the top-level domains extracted from the training set. The Large Random product corpus does not have this focus. Consequently, the number of hosts is a lot higher and the median number of records per host is lower.

Product Corpus	No. Records	No. Hosts	Median No. Records per Host	Max No. Records per Host
Small Related	75,248	1,160	100	400
Large Related	1,185,884	1,505	48	5,885
Large Random	1,029,063	98,421	2	2,878

Table 12: Size of Product Corpora

All extracted product offers have a title and at least one of the attributes description or category. The attributes are identified using the schema.org product annotations¹² name for title, description for description as well as category, breadcrumb and breadcrumbList for category. The attribute category is associated with multiple annotations, because different hosts use various annotations to categorise their products. Lastly, all attribute values are lowercased and excessive white spaces are removed. Table ?? shows that the product corpus Large Random has a comparably high percentage of missing description values. The Large Related product corpus has a lot of missing category values. These characteristics might influence the outcome of MLM.

Product Corpus	Median No. Characters Title	Median No. Characters Description	Missing Values Descrption	Median No. Characters Category	Missing Values Category
Small Related	38	310	10.43%	24	29.69%
Large Related	41	275	7.06%	22	68.90%
Large Random	34	39	72.99%	70	44.32%

Table 13: Distribution of Attributes in the Product Corpora

5.2 Attribute Combinations

For MLM the attributes title, category and description are used. The product categories do not follow the categories of the downstream hierarchical product classification, because these categories assigned by the online shops themselves. Still these categories can contain valuable product information. In the basic setup the attribute values are concatenated to a single line text representation of the product. This default attribute combination is referred to as Title-Cat-Desc. To measure the effect of the heterogeneous categories, two additional product text representations are used for MLM. In one scenario only title and description are considered for MLM. The categories are disregarded. This scenario is referred to as Title-Desc and encoded in the model’s name as <transformer model>_{nocat}. As alternative setup, the product attributes are split into two lines. One line contains

¹²<https://schema.org/Product>

the attribute values of title and category and the other line contains the attribute values of title and description. This scenario is referred to as Title-Cat/Title-Desc and encoded in the model’s name as <transformer model>_{ext}. This way the influence of the heterogeneous categories during language modelling can be measured. Due to the smaller size and the low percentage of missing category values of the product corpus Small Related as shown in Table ??, the impact of using the category information on the model performance is evaluated using this product corpus.

5.3 MLM Procedure

The pre-training used to inject knowledge about product offers into the RoBERTa_{base} model follows the MLM procedure used to pre-train RoBERTa_{base} initially. During MLM in each epoch a random sample of tokens from the input sequence is selected and replaced by the special token [MASK]. Uniformly 15% of the input tokens are selected for possible replacement. Of these selected tokens, 80% are replaced with [MASK], 10% are left unchanged and 10% are replaced by a randomly selected vocabulary token. For MLM a language modelling head predicts the masked tokens of the input. The MLM objective is a cross-entropy loss on predicting the masked tokens [? ?]. For the downstream hierarchical product classification the language modelling head is replaced by one of the task-specific classification heads introduced in Section ??.

5.4 Experimental Setup

For pre-training the RoBERTa_{base} models on the different product corpora, the chosen hyperparameters are a batch size of 4, a learning rate of 5e-5 and a linear weight decay of 0.01. All models are pre-trained for 5 epochs. The downstream hierarchical product classification follows the same settings as the baseline experiments described in Section ?. In this setting the average results on the test set over three randomly initialized runs for each experimental setup are reported. Based on their usefulness for the baseline models both attributes title and description are used as input for hierarchical product classification on the MWPDP task. For the Icecat/WDC222 task only the title is used as input. Since the collection of the product corpora focuses on the MWPDP task, the conducted experiments with an extended domain-specific MLM focus on the MWPDP task, too. The best performing pre-trained model on the MWPDP task is transferred to the Icecat/WDC222 task. Table ?? and Table ?? show the experimental results of an extended domain-specific MLM for hierarchical product classification. To reference the different models the same encoding as in Section ? is used. For <transformer model> the identifiers rel_s for pre-training on the Small Large corpus, rel_l for pre-training on the Related Large corpus and rand_l for pre-training on the Random Large corpus are added. Experiments with the same capital letter in the column "Same Error Rate" share the same error proportion on the test set according to McNemar’s significance test. Otherwise the experiment’s error proportion is significantly different.

5.5 Effect of Using Different Product Corpora

Table ?? shows that the baseline model 2-base-flat is outperformed by all other models on the MWPDP task. Our best model 2-rel_l-rnn outperforms the baseline model 2-base-flat by 1.22 wF1 and 1.18 hF1 points. According to the significance test this performance difference is significant. This demonstrates the positive impact of domain-specific MLM on hierarchical product classification. The performance increase of the model 2-rel_l-rnn compared to the models 2-rel_s-rnn and 2-rand_l indicates that a large number of related product offers improves the model’s performance the most. Among the classification heads, the results of the models 2-rel_l-flat, 2-rel_l-hier and 2-rel_l-rnn show that the RNN profits most from domain-specific pre-training. Table ?? reveals that the models 1-rel_l-rnn and 1-base-rnn have the same performance on the Icecat/WDC222 task. This underlines that the pre-training corpus has to be as similar as possible to the hierarchical product classification task to gain a significant performance boost from pre-training.

Model	Product Corpus MLM	Attribute Combination MLM	Head	wF1	Δ wF1	hF1	Δ hF1	Same Error Rate
2-base-flat	None	None	Flat	87.52		87.62	4.94	B
2-rel_l-flat	Large Related	Title-Cat-Desc	Flat	87.61	0.09	87.70	0.08	B
2-rel_s-rnn	Small Related	Title-Cat-Desc	RNN	88.31	0.79	88.47	0.85	C
2-rel_l-rnn	Large Related	Title-Cat-Desc	RNN	88.74	1.22	88.80	1.18	
2-rand_l-rnn	Large Random	Title-Cat-Desc	RNN	88.19	0.67	88.34	0.72	
2-rel_l-hierar.	Large Related	Title-Cat-Desc	Hierar.	88.44	0.92	88.60	0.98	
2-rel_s _{nocat} -rnn	Small Related	Title-Desc	RNN	88.27	0.75	88.41	0.79	C
2-rel_s _{ext} -rnn	Small Related	Title-Cat/ Title-Desc	RNN	88.74	1.22	88.98	1.36	

Table 14: Experimental results with Language Modelling - MWPD Task

Model	Product Corpus MLM	Attribute Combination MLM	Head	wF1	Δ wF1	hF1	Δ hF1	Same Error Rate
1-base-rnn	None	None	Flat	86.56		85.58		D
1-rel_l-rnn	Large Related	Title-Cat-Desc	RNN	86.38	-0.18	85.61	+0.03	D

Table 15: Experimental results with Language Modelling - Icecat/WDC222 Task

5.6 Effect of Using Web Shop Categories

The results in Table ?? indicate a slightly positive effect of using the heterogeneous categorization information from the original web shops during pre-training. A comparison of the models 2-rel_s_{nocat}-rnn and 2-rel_s-rnn shows that disregarding the web shop categories has a negative but not significant impact on the model’s performance. In the extended scenario of model 2-rel_s_{ext}-rnn, the model’s performance improves up to the performance level of the model 2-rel_l-rnn and significantly outperforms the baseline model 2-base-flat by 1.22 wF1 and 1.36 hF1 points on the MWPD task. A reason for these results might be that doubling the product representations during pre-training has a positive impact, because it almost doubles the amount of available text for pre-training. This effect is comparable to doubling the number of training epochs, which might improve the performance results, too. Another reason might be the length of the different attributes. The values of the attributes title and category are rather short compared to the attribute values of the description as shown by the median number of characters in Table ?. If the product offer is represented by a single line, the generated masked tokens during MLM are more likely part of the description than part of title or category. If mainly tokens from the description are masked, the model learns to better represent these long descriptions. At the same time, it can be assumed that the title is more informative than the lengthy description. By presenting the product offers twice with different attribute combinations to the model during pre-training the disturbing effect of long descriptions is reduced. This allows the model to better exploit the heterogeneous categories and the title. From our results we can conclude that the heterogeneous categories from the web shops have a slightly positive but not significant impact on the model’s performance.

6 Related Work

This section discusses related work on the domain adaptation of transformer models and gives an overview of the state of the art concerning hierarchical product classification using transformer models.

6.1 Domain Adaptation

The technique of pre-training transformer models on large text corpora and fine-tuning them for downstream tasks has proven successful in NLP [? ? ?]. BERT is one of these transformer models and was pre-trained on publicly available text corpora consisting such as the text of books and the English Wikipedia [?]. Through pre-training the model obtains the ability to encode natural language [?]. This knowledge about natural language is then transferred to downstream tasks. Pre-trained domain-specific models have shown that pre-training on domain-specific text improves the performance on downstream domain-specific tasks [? ? ? ?]. E-BERT for example uses adaptive masking on a product and a review corpus during pre-training to learn e-Commerce knowledge on phrase-level and on product-level. Pre-training enables E-BERT to outperform a BERT based model on different downstream tasks related to e-commerce [?]. Comparing the effects of adaptive masking and random masking using the product offer corpora that were created for this paper is an interesting direction for future work.

6.2 Hierarchical Product Classification

Related work shows that exploiting the hierarchical structure can improve classification results [? ? ? ? ?]. The participants of the MWPD challenge show that in addition to exploiting the hierarchy, pre-trained transformer models can boost the results of hierarchical product classification tasks [?]. Team Rhinobird, the winners of the MWPD challenge, combine a pre-trained transformer model BERT with a hierarchical classification head [?]. Their Dynamic Masked Softmax classification head sequentially predicts the categories of different levels in the product hierarchy by actively restricting the classes, which can be predicted on the lower levels based on the predicted parent level node. Through different BERT based representations an ensemble of classifiers with the Dynamic Masked Softmax head enables Rhinobird to reach a wF1 score of 88.08 on the MWPD task that is used in this paper. Additionally, Rhinobird [?] applies pseudo labelling on the unlabeled test data to further improve the performance of their model. This procedure might leak information about the test set into the training process. Thus, we compare our models to the Rhinobird results without pseudo labelling. Our best model based on a domain-specifically pre-trained RoBERTa model and a RNN classification head achieves a performance of 88.74 wF1 points. This is an improvement of +0.66 points over Rhinobird’s results. Given that Rhinobird achieves this good performance using an ensemble of models, future work could examine how an ensemble consisting of differently pre-trained and differently fine-tuned transformers can further improve the performance of our model. Team ASVInSpace uses a CNN based approach for language modelling with a multi-output classification head that predicts the categories of the different levels in the product hierarchy [?]. Our best model outperforms ASVInSpace’s approach by +2.14 wF1 points.

7 Conclusion

Our results show that the performance of transformer models on hierarchical product classification tasks can be improved through domain-specific pre-training on a corpus of related product offers. All domain-specifically pre-trained and fine-tuned models outperform the baseline model, which relies on general pre-training and task-specific fine-tuning. Our experiments with three different domain-specific corpora of product offers demonstrate that a large corpus of related product offers leads to the highest performance gain. If we adjust the product offer representation during pre-training to exploit the special characteristics of the attributes title, description and

category, the result on the hierarchical product classification task is further improved even though only a small corpus of related products is used for pre-training. With this approach and a task-specific classification head our best model outperforms the baseline model by 1.22 wF1 points and 1.36 hF1 points.

References

- [1] A. Joulin, E. Grave, P. Bojanowski and T. Mikolov. Bag of Tricks for Efficient Text Classification. *15th Conference of the European Chapter of the Association for Computational Linguistics*, 2:427–431, 2017.
- [2] J. Devlin, M. Chang, K. Lee and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 4171–4186, 2019.
- [3] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer and V. Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692 [cs]*, 2019.
- [4] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. So and J. Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36:1234–1240, 2019.
- [5] D. Zhang, Z. Yuan, Y. Liu, Z. Fu, F. Zhuang, P. Wang, H. Chen and H. Xiong. E-BERT: A Phrase and Product Knowledge Enhanced Language Model for E-commerce. *arXiv:2009.02835 [cs]*, 2020.
- [6] D. Gao, W. Yang, H. Zhou, Y. Wei, Y. Hu and H. Wang. Deep Hierarchical Classification for Category Prediction in E-commerce System. *3rd Workshop on e-Commerce and NLP (ECNLP)*, 64–68, 2020.
- [7] I. Beltagy, K. Lo and A. Cohan. SciBERT: A Pretrained Language Model for Scientific Text. *Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3615–3620, 2019.
- [8] C. Silla and A. Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22:31–72, 2011.
- [9] R. Meusel, A. Primpeli, C. Meilicke, H. Paulheim and C. Bizer. Exploiting Microdata Annotations to Consistently Categorize Product Offers at Web Scale. *International Conference on Electronic Commerce and Web Technologies*, 83–99, 2015.
- [10] S. Kiritchenko, S. Matwin and A. Famili. Functional Annotation of Genes Using Hierarchical Text Categorization. *ACL Workshop on Linking Biological Literature, Ontologies and Databases: Mining Biological Semantics*, 2005.
- [11] Z. Zhang, C. Bizer, R. Peeters and A. Primpeli. MWPD2020: Semantic Web challenge on Mining the Web of HTML-embedded product data. *CEUR Workshop Mining the Web of HTML-embedded Product Data*, 2720:2–18, 2020.
- [12] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov and Q. Le. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *33rd Conference on Neural Information Processing Systems (NeurIPS)*, 2020.
- [13] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li and P. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Machine Learning Research*, 21:1–67, 2020.
- [14] L. Yang, E. Shijia, S. Xu and Y. Xiang. Bert with Dynamic Masked Softmax and Pseudo Labeling for Hierarchical Product Classification. *CEUR Workshop Mining the Web of HTML-embedded Product Data*, 2720:2020.
- [15] J. Wehrmann, R. Cerri and R. Barros. Hierarchical Multi-Label Classification Networks. *35th International Conference on Machine Learning*, 5075–5084, 2018.
- [16] R. You, Z. Zhang, Z. Wang, S. Dai, H. Mamitsuka and S. Zhu. AttentionXML: Label Tree-based Attention-Aware Deep Model for High-Performance Extreme Multi-Label Text Classification. *33rd Conference on Neural Information Processing Systems (NeurIPS)*, 32:11, 2019.
- [17] Z. Zhang and M. Paramita. Product Classification Using Microdata Annotations. *International Semantic Web Conference (ISWC)*, 716–732, 2019.
- [18] J. Borst, E. Krner and A. Niekler. Language Model CNN-driven similarity matching and classification for HTML-embedded Product Data. *CEUR Workshop Mining the Web of HTML-embedded Product Data*, 2720:11, 2020.
- [19] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou and H. Hon. Unified Language Model Pre-training for Natural Language Understanding and Generation. *33rd Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

- [20] A. Primpeli, R. Peeters and C. Bizer. The WDC Training Dataset and Gold Standard for Large-Scale Product Matching. *International World Wide Web Conference (WWW)*, 381–386, 2019.
- [21] T. Dietterich. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10:1895–1923, 1998.
- [22] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma and R. Soricut. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *International Conference on Learning Representations (ICLR)*, 2020.
- [23] S. Gururangan, A. MarasoviÄ, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey and N. Smith. Don’t Stop Pretraining: Adapt Language Models to Domains and Tasks. *58th Annual Meeting of the Association for Computational Linguistics*, 8342–8360, 2020.

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

Xu Liu¹, Congzhe Su², Amey Barapatre², Xiaoting Zhao², Diane Hu², Chu-Cheng Hsieh², Jingrui He³

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 [? ? ? ? ?].

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 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 “wristwate” must produce



figs/fig_ranking_system2.pdf

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.

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 ??): 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 [? ? ?]; 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) [? ? ? ?]. 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 *Online Personalized Attribute-based Re-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 [?], product category [? ?] or a context [? ? ?]. 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 ?? 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 ??). 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 [? ?]. Significant

breakthroughs in deep learning (i.e, batch normalization and dropout), have led to its wide adoptions in various communities and applications [?]. In [?], 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 [? ? ? ? ?].

Given that a long-term memory models are insufficient to address drift in user interests, [?] 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, [?] 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, [?] 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, [?] 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 [?] proposed a Matrix Factorization-based attention model to address large-volume and high-velocity session streaming data and [?] 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 [?] 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 [?], Upper Confidence Bounds [?], Thompson sampling [?], EXP3 [?], and others [?]. In the e-commerce [?] 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 [?] 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 [?] 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 [?] 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. [?] proposed an interactive collaborative topic regression model that infers the clusters of arms via topic models [?] and then utilizes dependent arms for the recommendation.

In this literature, it is common to address this problem by treating each arm in the bandit to represent a single item [?], product category [?] or a context [? ? ?]. 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



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

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 :

$$A_t(x_i) = \begin{cases} 0, & \text{no action on } x_i \\ 1, & x_i \text{ is purchased} \\ 2, & x_i \text{ is added to cart} \\ 3, & x_i \text{ is clicked} \end{cases}, \forall x_i \in I_t. \quad (24)$$

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 : $x_i = \{atr_1, atr_2, \dots, atr_{H_{x_i}}\}$.

Figure ?? 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 Θ on attributes as beta-distributed: $\Theta = \{\theta_{atr_n}\}_{n=1}^N \sim \{Beta(\alpha_{atr_n}, \beta_{atr_n})\}_{n=1}^N$, 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 [?], let θ_{atr_n} be beta-distributed, with $\alpha_{atr_n}, \beta_{atr_n}$ be the two parameters of the distribution. In Section ?? 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 Θ . 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 Θ , 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 Θ to optimize in-session personalization?

Input: At time t , (1) Candidate list of items I_t , and (2) user in-session preference Θ .

Output: Sequentially learn $f_t : I_t \times \Theta \rightarrow \tilde{I}_t$.

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.

[!t]

Given a session $S = \{[Q_t, I_t, A_t]\}_{t=1}^T$

$\{\delta_i\}_{i: \text{actions}}$: action-aware increments on attribute parameters

γ : 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

$[Q_t, I_t, A_t] \in S$ **(1) Rerank on the Item List** $f : I_t \rightarrow \tilde{I}_t$

sample $s_{atr_h} \sim Beta(\alpha_{atr_h}, \beta_{atr_h}), \forall atr_h \in N_S$

$x_i \in I_t$ Given $x_i = \{atr_h\}_{h=1}^{H_{x_i}}$ as associated attributes in x_i , set $score(x_i) = \sum_{atr_h \in x_i} g(s_{atr_h})$ $\tilde{I}_t = sorted([score(x_i)]_{x_i \in I_t})$

(2) Update attribute parameters given A_t

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

Let $\mathcal{V}_t = \cup\{atr_h : \forall atr_h \in x_i \forall x_i \in I_t\}$

$x_i \in I_t$ $A_t(x_i) \neq 0$, item x_i has positive actions $\alpha_{atr_h} + = \delta_{A_t(x_i)} \times \{1 - Exp(-|\mathcal{U}_t|_0)\}, \forall atr_h \in x_i$

$A_t(x_i) = 0$, no action on item x_i $\beta_{atr_h} + = \delta_{A_t(x_i)} \times \{1 - Exp(-\gamma|\mathcal{V}_t \setminus \mathcal{U}_t|_0)\}, \forall atr_h \in x_i$ **end** All

re-ranking results $[\tilde{I}_t]_{t=1}^T$

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 [?].

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 :

$$\theta_{atr_h} \sim Beta(\alpha_{atr_h}, \beta_{atr_h}). \quad (25)$$

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

$$score(x_i) = \sum_{atr_h \in x_i} g(\theta_{atr_h}), \quad (26)$$

where $g(\theta_{atr_h}) = \frac{1}{rank(\theta_{atr_h})}$ is a harmonic 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 satisfaction 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 $[score(x_i)]_{x_i \in I_t}$.

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 $x_i \in I_t$:

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

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,

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

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

$$\begin{aligned} \alpha_{atr_h} + &= \sum_{\tilde{\mathcal{Y}}_{t,atr_h}} \delta_{A_t(x_i)} \left(1 - e^{-|\mathcal{U}_t|_0}\right), \forall atr_h \in \mathcal{U}_t, \\ \beta_{atr_h} + &= \sum_{\tilde{\mathcal{Z}}_{t,atr_h}} \delta_{A_t(x_i)} \left(1 - e^{-\gamma|\mathcal{V}_t \setminus \mathcal{U}_t|_0}\right), \forall atr_h \in \mathcal{V}_t \setminus \mathcal{U}_t, \end{aligned} \quad (27)$$

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

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

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.

Table 16: 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

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 ??*)

Experiment #2: How does *OPAR* perform as an action-aware model? (*Answered in subsection ??*)

Experiment #3: How does *OPAR* help to understand users’ short-term, in-session shopping preference? (*Answered in subsection ??*)

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 ?? 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 ?. The remaining data is the **testing dataset**, on which we report re-ranking performance for *OPAR* and other baseline algorithms on in Table ??.

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 ??). 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) [?], 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:

$$\text{click-NDCG}_t = \text{click-DCG}_t / \text{IDCG}_t, \forall t = 1, \dots, T, \quad (28)$$

and *click-NDCG* of a session S is the average of *click-NDCG_t* over events that have at least one click:

$$\text{click-NDCG} = \text{Average}(\text{click-NDCG}_t). \quad (29)$$

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 ??, 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 [?] is the boosted tree version of LambdaRank [?], 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.
2. Atr-KNN is derived from Item-KNN [?]. 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.

Table 17: Re-ranking performance comparison on over all data sets (top-left) and 7 category-specific data sets.

		Over All Category						Clothing					
		LambdaMART	Atr-KNN	Atr-POP	GRU4Rec	OPAR	OPAR _w	LambdaMART	Atr-KNN	Atr-POP	GRU4Rec	OPAR	OPAR _w
4* Purchase NDCG	@4	0.1795	0.0130	0.0749	0.0618	0.2994	0.3042	0.1948	0.0103	0.0516	0.0551	0.2384	0.2494
	2-15 @12	0.2629	0.0412	0.1323	0.1425	0.3505	0.3607	0.2670	0.0348	0.1269	0.0824	0.2685	0.2744
	2-15 @24	0.3162	0.1260	0.2112	0.2018	0.3718	0.3900	0.3019	0.0090	0.2193	0.1434	0.3209	0.3263
	2-15 @48	0.3724	0.2554	0.2861	0.2518	0.4512	0.4578	0.3774	0.2462	0.2784	0.2157	0.3976	0.4030
4* Click NDCG	@4	0.1459	0.0816	0.0705	0.0701	0.3120	0.3158	0.1328	0.0067	0.0690	0.0691	0.3058	0.3197
	2-15 @12	0.2265	0.1456	0.1264	0.1354	0.3213	0.3229	0.2137	0.0228	0.1224	0.1414	0.3126	0.3257
	2-15 @24	0.2955	0.2157	0.2021	0.1922	0.3318	0.3489	0.2821	0.0658	0.2045	0.1844	0.3274	0.3424
	2-15 @48	0.3815	0.3245	0.2813	0.2689	0.4047	0.4051	0.3711	0.2309	0.2807	0.2613	0.3988	0.4061
		Home & Living						Paper & Party Supplies					
		LambdaMART	Atr-KNN	Atr-POP	GRU4Rec	OPAR	OPAR _w	LambdaMART	Atr-KNN	Atr-POP	GRU4Rec	OPAR	OPAR _w
4* Purchase NDCG	@4	0.1755	0.0131	0.0649	0.0571	0.2920	0.2952	0.1822	0.0010	0.1255	0.0684	0.2828	0.2965
	2-15 @12	0.2670	0.0396	0.1226	0.1281	0.3391	0.3436	0.2667	0.0406	0.1692	0.0941	0.3367	0.3497
	2-15 @24	0.3218	0.0936	0.2066	0.1752	0.3838	0.3879	0.3276	0.1297	0.2469	0.1542	0.3796	0.3905
	2-15 @48	0.3874	0.2543	0.2789	0.2164	0.4462	0.4491	0.3876	0.2550	0.3216	0.1943	0.4291	0.4399
4* Click NDCG	@4	0.1481	0.0054	0.0601	0.0944	0.3201	0.3219	0.1585	0.0052	0.1084	0.0839	0.2825	0.2874
	2-15 @12	0.2294	0.0213	0.1175	0.1416	0.3244	0.3256	0.2394	0.0247	0.1586	0.1367	0.2931	0.2973
	2-15 @24	0.2978	0.0598	0.1973	0.1843	0.3485	0.3491	0.3103	0.0644	0.2300	0.1742	0.3383	0.3189
	2-15 @48	0.3835	0.2278	0.2746	0.2288	0.4032	0.4086	0.3911	0.2306	0.3104	0.2007	0.4017	0.4072
		Craft Supplies & Tools						Accessories					
		LambdaMART	Atr-KNN	Atr-POP	GRU4Rec	OPAR	OPAR _w	LambdaMART	Atr-KNN	Atr-POP	GRU4Rec	OPAR	OPAR _w
4* Purchase NDCG	@4	0.1912	0.0135	0.0739	0.0741	0.3101	0.3268	0.1954	0.0251	0.0683	0.0511	0.2166	0.2178
	2-15 @12	0.2735	0.0407	0.1296	0.1125	0.3673	0.3781	0.2828	0.0741	0.1431	0.0849	0.2835	0.2930
	2-15 @24	0.3272	0.1208	0.1970	0.1644	0.4084	0.4188	0.3324	0.1406	0.2510	0.1222	0.3304	0.3361
	2-15 @48	0.3844	0.2577	0.2820	0.2214	0.4366	0.4750	0.3869	0.2693	0.2917	0.1641	0.3962	0.4020
4* Click NDCG	@4	0.1458	0.0055	0.0749	0.0994	0.3118	0.3166	0.1502	0.0105	0.0673	0.0712	0.2495	0.2605
	2-15 @12	0.2262	0.0513	0.1290	0.1279	0.3241	0.3293	0.2324	0.0439	0.1358	0.1331	0.2656	0.2708
	2-15 @24	0.2955	0.2042	0.1953	0.1935	0.3525	0.3521	0.3006	0.1091	0.2398	0.1800	0.3155	0.3212
	2-15 @48	0.3811	0.2278	0.2815	0.2277	0.4080	0.4078	0.3848	0.2391	0.2885	0.2312	0.3548	0.4029
		Electronics & Accessories						Jewelry					
		LambdaMART	Atr-KNN	Atr-POP	GRU4Rec	OPAR	OPAR _w	LambdaMART	Atr-KNN	Atr-POP	GRU4Rec	OPAR	OPAR _w
4* Purchase NDCG	@4	0.2136	0.0501	0.0715	0.0814	0.2847	0.2995	0.1661	0.0060	0.0576	0.0718	0.3051	0.3285
	2-15 @12	0.3014	0.1109	0.1546	0.1223	0.3386	0.3782	0.2534	0.0766	0.1074	0.1142	0.3484	0.3854
	2-15 @24	0.3519	0.0176	0.2652	0.1674	0.4257	0.4152	0.3087	0.1470	0.1668	0.1847	0.3866	0.3973
	2-15 @48	0.4060	0.2965	0.2981	0.2416	0.4516	0.4656	0.3814	0.2460	0.2663	0.2367	0.4425	0.4598
4* Click NDCG	@4	0.1530	0.0267	0.0805	0.0641	0.2074	0.2051	0.0701	0.0027	0.0621	0.0614	0.3314	0.3892
	2-15 @12	0.2324	0.0703	0.1580	0.0939	0.2487	0.2622	0.1314	0.0106	0.1141	0.1021	0.3783	0.3963
	2-15 @24	0.3029	0.1410	0.2657	0.1345	0.3158	0.3120	0.1989	0.1276	0.1762	0.1647	0.3956	0.4162
	2-15 @48	0.3880	0.2560	0.3026	0.1667	0.3978	0.4078	0.3119	0.2192	0.2700	0.2144	0.4190	0.4475

- 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 [?] given its simplicity and efficacy.
- GRU4Rec [?] 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.

Table 18: Multiple Purchase Intents within One Session

Timestamp		Query	Query Taxonomy		Engaged Attributes	
2-6 4* 1st Purchase Intent	0	'flower girl basket'	paper and party supplies	(NO ACTION)	Browsing	
	2-6	1-4	'flower girl basket wedding'	paper and party supplies	(CLICK)	'Prime Color: White', 'Occasion: Wedding', 'Holiday: Christmas', 'Wedding theme: Beach & tropical', 'Craft type: Floral arranging'
	2-6	5-9	'flower girl basket beach wedding'	paper and party supplies	(CLICK)	'Prime Color: Blue', 'Occasion: Wedding', 'Holiday: Christmas', 'Wedding theme: Beach & tropical', 'Secondary color: White', 'Craft type: Floral arranging'
	2-6	10-11	'two flower girl and one pillow'	paper and party supplies		Browsing
	2-6	Purchase Intent Change				
2-6 3* 2nd Purchase Intent	12-15	'hat for beach wedding'	clothing.women_clothing	(CLICK)	'Prime Color: Blue', 'Occasion: Wedding'	
	16-22	'turquoise petals'	accessories	(CLICK)	'Prime Color: Blue', 'occasion: Bridal shower', 'Wedding theme: Fairytale & princess'	
	2-6	23	'bride hair decoration beach theme'	clothing.women_clothing	(NO ACTION)	Browsing
	2-6	24	'turquoise petals'	accessories	(PURCHASE)	'Prime Color: Blue', 'Occasion: Bridal shower', 'Wedding theme: Fairytale & princess'
2-6 2* Final Purchase						

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 ?? shows experiment results of our model (*OPARs*) against 4 baselines described in Section ?. 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 achieved 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.

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 ?? 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 ??, 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 ?? presents a record of a user's in-session activities. Figure ?? (top) shows the sequential improvement of *OPAR* in session-level click-NDCG over time compared to the baseline, and Figure ?? (bottom) shows how *OPAR*

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

captures user’s preference, θ_{atr_h} , on 5 attributes over time. The “Engaged Attributes” column in Table ?? maps out all attributes associated with the clicked items for the corresponding query.

As shown in Table ??, 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 $t = 5, \dots, 9$ 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 ?? (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 & 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 & 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. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & 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 & Data Mining, 2019.
- [22] P. Pobrotyn, T. Bartczak, M. Synowiec, R. Białobrzewski, 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 & Data Mining*, 2020.



Data Engineering

It's FREE to join!

TCDE

tab.computer.org/tcde/

The Technical Committee on Data Engineering (TCDE) of the IEEE Computer Society is concerned with the role of data in the design, development, management and utilization of information systems.

- Data Management Systems and Modern Hardware/Software Platforms
- Data Models, Data Integration, Semantics and Data Quality
- Spatial, Temporal, Graph, Scientific, Statistical and Multimedia Databases
- Data Mining, Data Warehousing, and OLAP
- Big Data, Streams and Clouds
- Information Management, Distribution, Mobility, and the WWW
- Data Security, Privacy and Trust
- Performance, Experiments, and Analysis of Data Systems

The TCDE sponsors the International Conference on Data Engineering (ICDE). It publishes a quarterly newsletter, the Data Engineering Bulletin. If you are a member of the IEEE Computer Society, you may join the TCDE and receive copies of the Data Engineering Bulletin without cost. There are approximately 1000 members of the TCDE.

Join TCDE via Online or Fax

ONLINE: Follow the instructions on this page:

www.computer.org/portal/web/tandc/joinatc

FAX: Complete your details and fax this form to **+61-7-3365 3248**

Name

IEEE Member #

Mailing Address

Country

Email

Phone

TCDE Mailing List

TCDE will occasionally email announcements, and other opportunities available for members. This mailing list will be used only for this purpose.

Membership Questions?

Xiaoyong Du

Key Laboratory of Data Engineering
and Knowledge Engineering
Renmin University of China
Beijing 100872, China
duyong@ruc.edu.cn

TCDE Chair

Xiaofang Zhou

School of Information Technology and
Electrical Engineering
The University of Queensland
Brisbane, QLD 4072, Australia
zxf@uq.edu.au

IEEE Computer Society
10662 Los Vaqueros Circle
Los Alamitos, CA 90720-1314

Non-profit Org.
U.S. Postage
PAID
Los Alamitos, CA
Permit 1398