Addressing Cold Start in Product Search via Empirical Bayes

Cuize Han
Amazon, Palo Alto, CA, USA

Pablo Castells
Amazon, Madrid, Spain

Parth Gupta
Amazon, Palo Alto, CA, USA

Xu Xu
Amazon, Palo Alto, CA, USA

Vamsi Salaka
Amazon, Palo Alto, CA, USA

ABSTRACT

Cold start is a challenge in product search. Profuse literature addresses related problems such as bias and diversity in search, and cold start is a classic topic in recommender systems research. While search cold start might be seen conceptually as a particular case in such areas, we find that available solutions fail to specifically and practically solve the cold-start problem in product search. The problem is complex as exposing new products may come at the expense of primary business metrics (e.g. revenue), and involves a complex balance between customer satisfaction, seller satisfaction, business performance, short-term gains and long-term value.

In this paper, we propose a principled approach to deal with cold start in a large-scale e-commerce search system. We discuss how product ranking is affected by non-behavioral topical relevance and behavioral popularity, and their role in introducing biases that result in cold-start for ranking new products. Our approach applies Empirical Bayes to model behavioral information via non-behavioral signals in terms of priors, and effectively estimate true engagement posterior updates. We report comprehensive offline and online experiments over large datasets that show the effectiveness of our methods to address cold start, and provide further insights. An online A/B test on 50 million queries shows a significant improvement in new product impressions by 13.53% and a significant increase in new product purchase by 11.14%, with overall purchases up by 0.08%, highlighting the empirical effectiveness of the approach.

CCS CONCEPTS

• **Applied computing** → **Online shopping**; • **Information systems** → **Novelty in information retrieval**.

KEYWORDS

Cold start, discovery, exploration, bias, feedback loop, empirical Bayes, e-commerce search

ACM Reference Format:
Cuize Han, Pablo Castells, Parth Gupta, Xu Xu, and Vamsi Salaka. 2022. Addressing Cold Start in Product Search via Empirical Bayes. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3511808.3557066

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557066

1 INTRODUCTION

E-commerce search plays a critical role in connecting the specific needs of customers with relevant products. At the same time, it is a service connecting sellers with buyers. A growing e-commerce website continuously onboards new customers, sellers and sees many new products being listed every week. Modern search systems employ learning-to-rank (LTR) solutions to rank relevant products against customer queries. In such systems, many LTR features – the most effective ones – are based on users' interactions with products, such as impressions, clicks, and purchases [19, 47]. Ranking models are trained to optimize user engagement, and therefore, such behavioral features tend to be the most important training signals [49].

However, new and tail products with missing or sparse user engagement lack reliable behavioral features, and are hence ranked as irrelevant, which further excludes them in turn from catching up with user engagement. It takes a long time for such items to gather enough behavioral signals to show up at their fair ranking position. This leads to the causality dilemma: no behavioral data causes poor ranking which in turn results in new products having a reduced likelihood of accruing behavioral data. This phenomenon is referred to as the *cold start* problem and poses serious concerns, from poor customer experience to seller frustration and lost revenue opportunity.

New products suffer from various biases in e-commerce search [32]: i) item selection bias: products with strong behavioral features are returned again and again in response to queries, and customers engage with them, forming a reinforcing loop that creates a bias against new products; ii) position bias: even when new products make it to the result set, they are likely to be ranked at low positions, that users are less likely to examine; and iii) trust bias: users trust well-known brands and rely on customer reviews, so they engage less with new products lacking them. Solutions for cold start in e-commerce should explore new products related to customer queries even if they lack observed interaction [14].

However, the problem is challenging as exposing new products may come at the expense of key business metrics (e.g. revenue), and involves a complex balance between customer satisfaction, seller satisfaction, business performance, short-term gains and long-term value. On one hand, over-exploration of new products can hurt the business metrics while exploring irrelevant or bad quality products can damage customer trust. On the other hand, new product exploration is beneficial for the long-term health of the marketplace, provides a wider selection to customers, and encourages the incorporation of new sellers and therefore growth of business.

In this paper, we propose a principled approach to deal with cold start in a large-scale e-commerce search system. We first lay out a framework where we consider how product ranking is affected by non-behavioral topical relevance and behavioral popularity, and their role in the feedback loop that introduces biases resulting in cold start for ranking new products. We analyze the root of the

cold-start problem in i) the lack of generalization power of non-behavioral features, requiring the help from behavioral features' memorization power; and ii) the naive use of behavioral features, bringing significant bias against cold-start items.

Upon this view, we present an approach to deal with cold start while preserving a good tradeoff with key business metrics. Our approach applies Empirical Bayes to model behavioral information via non-behavioral signals in terms of priors, and effectively estimate true engagement posteriors. We report comprehensive offline and online experiments in a large-scale e-commerce search system that show the effectiveness of our methods to address cold start, and provide further insights. An online A/B test on 50 million queries shows a significant improvement in new product impressions by 13.53% and a significant increase in new product purchase by 11.14%, with overall purchases up by 0.08%, highlighting the empirical effectiveness of the approach.

The paper is organized as follows: we start with an overview and discussion of cold start in search and recommendation and related work in Section 2. The framework to formalize the challenges and define the components of a working system handling cold start is presented in Section 3. We then present our specific approach, based on online feature exploration via Empirical Bayes in Section 4. We report experimental results with a simulated and a real search system in Sections 5 and 6. Following this, we present in Section 7 an enhanced version of our system that achieves improved online results. We end with some concluding remarks in Section 8.

2 BACKGROUND AND RELATED WORK

Modern e-commerce search engines address product selection and ranking as a supervised learning to rank task [41, 43]. The inputs for ranking models include product features, query features, and product-query features. Features can be further categorized into behavioral and non-behavioral. The latter are derived from intrinsic properties of products and queries, such as product category, brand, price or delivery speed, as well as associated textual content (title, about, description, etc.), images, video, etc.; and query properties such as length, specificity or categorization. Functional features relating products and queries, such as classic text IR models (e.g. tf-idf and BM25), are particularly effective in this scope.

Behavioral features, on their side, get created as the result of the interaction between customers and products through impressions, clicks, purchases, and other actions. Customer-product interaction records are also used as labels for learning to rank model training, under the understanding that no one is better placed than customers themselves to tell which products are relevant for a query [41], particularly so when voting with their wallet. For this reason, product-query behavioral features are the ones the ranking models rely on the most when ranking products: a product that semantically matches a query is not relevant if the customers who enter the search do not want to buy the product [49].

With the importance of behavioral features, and the role of user-item interaction in the learning objective, e-commerce search can be seen, in a way, as a hybrid system involving both search and recommendation [48]: semantic matching certainly plays a part in selecting relevant search results (particularly an initial candidate set), but customer-query-product interaction records typically have a heavier role in obtaining a boost in search relevance and business performance. As a consequence, e-commerce search is greatly impacted by cold start compared to search engines in other domains, such as web search, where semantic features may play a more prominent part.

New items typically cross the cold start barrier through very specific queries where a small result set size gives them a chance to be exposed to interested customers with a narrow, very well specified need. From that point on, items get a chance to hopefully emerge from behavioral starvation upon the initial attention spark. But this process is slow and venturesome, as it relies on sparse events and a non-negligible degree of luck. Sponsored results, listings and promotions are also essential instruments for sellers to push their new products into the search feedback wheel. Such mechanisms are however collateral to result search relevance, involve cost and effort for sellers, and a challenging competition that is not easy to overcome. With a fast stream of incoming new products in large-scale online markets, cold start becomes a pressing problem and a substantial source of missed opportunity for shoppers, merchants and the service [14].

2.1 Cold start in search

Work specifically dealing with search cold start is not easy to find in the literature. Haldar et al. [15], for instance, report experiences in dealing with cold start in the Airbnb search system. The explored approaches include the injection of new items in search results. Strategies in this line are also described by Taank et al. in a Google patent [42]. Haldar et al. describe unsolved tradeoffs and complexities involved in this approach. The authors were able to draw some improvement by training customer engagement predictors in the absence of observed customer feedback. More recently, Gupta et al. [14] developed a similar approach, where the engagement prediction was used both to smooth down variance in observed interaction, and to spark an initial engagement signal that would help new items emerge from the initial absence of signal. Missault et al. [29] discussed transfer learning strategies to train search models in new domains where customer feedback is not yet available.

While such work is highly relevant for our current purpose, we find that further efforts are needed to continue, grow and improve the results achieved in these experiences, better address the involved tradeoff costs, produce more effective and generalizable practical solutions, and develop a deeper understanding of the cold-start problems. Beyond our specific focus of concern, many related efforts address parts of the challenges or similar ones in a different context. We discuss them next for comparison and motivation.

2.2 Cold start in recommendation

Under the connection between e-commerce search and recommendation we drew in the previous section [48], it would be natural to tap on the long strand of research and practice in dealing with recommendation cold start. Cold start is indeed a classic topic in recommender systems research, as it is the obvious Achilles' heel of collaborative filtering, since cold start in this context means the absence of input that such methods feed on [1]. Solutions generally consist in using side information to make up for the lack of interaction data. Side information typically includes item-specific features [25, 30, 37, 45, 51], and/or user-features [6, 12], and/or social information [39, 40]. In essence, and as a gross simplification, these approaches can be seen as finding similarities between items (or between users) and transferring the patterns learned by the system on "warm" items onto new and cold items.

However, e-commerce search generally has heavier requirements and customer expectations on topical relevance than recommendation has, and novelty-accuracy tradeoffs that recommendation can bear are often not affordable in search: customers may put up with the occasional irrelevant recommendation in exchange for other valuable discoveries, but returning irrelevant search results heavily erodes customer trust and reliance on the service as a whole. New products that very few customers have interacted with incur a high risk of presenting results that no one will purchase, and detract from the immediate business value that other products of proven interest would procure. A direct application of recommendation cold-start methods "as are" to e-commerce search may therefore not result in an appropriate balance when the customer is searching with a specific need and purpose, under the tight requirements of a sensitive online business.

2.3 Addressing bias in learning to rank

Learning to rank (LTR) is the application of machine learning for solving ranking problems in information retrieval systems. Using expert annotations as training labels, a scoring function for ranking is learned that can generalize to predict relevance based on query and document features [28]. Many popular LTR algorithms such as LambdaRank/MART [7] and neural learning to rank [13] have been proposed and adopted by industry, achieving substantial improvements in retrieval effectiveness.

However, expert annotations are expensive and time-consuming to obtain [9]. Furthermore, in many applications, such as e-commerce search ranking, user preferences are changing over time and are often misaligned with experts' opinion [24, 36]. On the other hand, the implicit feedback from users when they are interacting with the system is cheap to obtain and directly reflects user preferences. But they are also very noisy and biased [10]. Thus important research efforts have been invested into unbiased learning to rank techniques in recent years, aiming to learn unbiased user preferences from biased user interactions. Two main families of methods can be distinguished in this direction: counterfactual learning to rank and online learning to rank.

2.3.1 Counterfactual learning to rank. In this perspective, a ranker is learned offline using implicit feedback collected from a production system (logging ranker). Methods in this family focus on both capturing interaction biases and optimization methods that can correct for them. Counterfactual LTR methods seek to extract the influence of the logging policy out of the user actions, and obtain an estimate of the true relevance of search results based on implicit feedback. This is achieved by modeling customer behaviors (usually by probabilistic click models), where the model parameters are estimated and determined by separate experiments[5]. After user observation probability (known as *propensity*) is estimated, counterfactual risk minimization is applied for model training [20], and counterfactual evaluation is applied in computing offline metrics. The loss functions and offline metrics are debiased from the logging policy, and weighted by the current training policy, which is more aligned with the online metrics that we care about.

Counterfactual LTR can optimize neural networks and DCG-like methods through upper-bounding [2, 4]. In particular, LambdaMART can also be adapted to counterfactual LTR [17]. In proposed approaches, inverse propensity weighting is a popular strategy that is often used to mitigate position bias [2, 20]. Counterfactual LTR can address trust bias [3] and item-selection bias [34, 46] as well. However, counterfactual LTR can only deal with bias in observed labels and only works when non-behavioral features are strong enough to predict the target. For instance, a reasonable ranking model can be produced by using just non-behavioral features training on debiased labels. But this approach cannot solve the new challenge in behavioral features.

2.3.2 Online learning to rank. In this formulation, rankers are interactively optimized under a stream of user interactions, rather than from a set of labeled data. This is fundamentally different from currently dominant supervised learning to rank approaches for information retrieval. Rather than explicitly modeling customer behavior and estimating click models, online LTR methods perturb the ranker parameters in random directions and select the most promising candidates for update through interleaving or online evaluation [16, 31, 38, 50].

Online LTR falls short however in addressing the new challenge in behavioral features. Like counterfactual LTR, it only deals with bias in labels, and relies on the assumption of a strong predictive power of non-behavioral features. Also, it cannot update the ranker very often in practice. Furthermore, online LTR finds challenges of its own in an e-commerce environment, such as the concept drift involved in day-of-week effects (and seasonality in general [14]), delayed customer feedback (users often close purchase decisions days after searching), and other aspects making this a noisy environment, with which a sufficiently dynamic and robust online learning solution is difficult to accommodate.

3 FRAMEWORK

We now formalize the problem setup and describe the framework for Learning and Memorizing to Rank (LMTR). Notations are summarized in table 1. This setting is generally applicable to many real-world search recommendation tasks, such as e-commerce product search or video search and recommendation. We find LMTR to be an important yet under-explored problem in the information retrieval field.

3.1 Relevance and popularity

A user issues a query q_t at time t to find a desired item. In response to the query, the search system returns a set of items, the "match set" D_{q_t}, and displays it as a ranked list $\pi_t : d \in D_{q_t} \rightarrow \{1, 2, ..., |D_{q_t}|\}$ according to a scoring function $f_t : D_{q_t} \times Q \rightarrow \mathbb{R}$ which we also refer to as a ranking function (Q being the space of all queries). Users then browse the delivered results list, producing explicit feedback (clicks, add to cart, add to favorites, purchase, etc.) as they engage with the system. For simplicity, we assume the feedback consists of clicks and we represent this feedback as a binary vector $\boldsymbol{c}_t \in \{0, 1\}^{|D_{q_t}|}$. Given this setting, the goal is to learn scoring functions f_t such that the expected cumulative reward over a period of time T, $\mathbb{E}\left[\sum_{t=0}^{T} |\boldsymbol{c}_t|_1\right] = \sum_{t=0}^{T} \sum_{d \in D_{q_t}} \mathbb{E}\left[c_{\pi_t(d)}\right]$ is maximized, where $|\cdot|_1$ denotes the L^1 norm.

User feedback can be modeled as a random vector that depends on the user's browsing behavior when examining the list, and the attractiveness of the items with respect to the query for the user. This is generally referred to as a click model [10]. The click probability can be expressed in terms of the examination probability and the attractiveness probability. The former can be modeled as an affine function of the position, accounting for both position and trust bias

Table 1: Notation summary.

q_t	Query issued at time t
D_{q_t}	Match set of items under q_t
f_t	Scoring function (ranker) at time t
π_t	The ranked list based on f_t
$\boldsymbol{c}_t$	A vector indicating whether a item in D_{q_t} got clicks
A	Attractiveness of an item under a query
$\boldsymbol{x}^{nb}$	Non-behavioral feature vector of a query item pair
$\boldsymbol{x}_t^b$	Behavioral feature vector of a query item pair at time t
ρ	Prediction (generalization) power of X^{nb} on A
$\mathcal{D}$	Customer interaction data (behavioral information)
ϕ	Function that calculates behavioral features $\boldsymbol{x}_t^b$ from $\mathcal{D}$

[44]. Click data can be debiased upon an estimate of the parameters in this affine function (e.g. based on historical A/B test data [5]).

The attractiveness component, on the other hand, is commonly handled in existing debiasing approaches as a simplified issue of unidimensional relevance. We contend that more effective approaches can be developed by considering two driving elements that determine attractiveness: a) the topical or semantic relevance between queries and items, and b) the popularity of the items among the users issuing the queries.

It is natural to assume that topical relevance can be well captured by content-based item features such as the product title and description, and the query keywords entered by customers. However, the popularity of items for queries can hardly be captured by content-based features only [15]. Based on our experience in e-commerce search, even large tree ensembles or sophisticated deep-learned embeddings underperform in predicting popularity from content information. This is not surprising: shoppers select search results not just based on how well they match the query, but also based on their beliefs and expectations of product quality, built upon past experience, other customers' reviews, and an assessment of how the product may please and fit the user's needs. This type of information is not easy to capture in query words and product descriptions alone.

3.2 Behavioral and non-behavioral features

Given a query q and an item d, we denote non-behavioral features as $\boldsymbol{x}^{nb} = \boldsymbol{x}^{nb}(q, d)$ and behavioral features as $\boldsymbol{x}^{b} = \boldsymbol{x}^{b}(q, d)$. Behavioral features can potentially depend on users, but we do not consider personalization here for simplicity. Behavioral features capture historical engagement (to simplify, click) statistics on query-item pairs, or pooled statistics of historical clicks on groups of query-item pairs.

Behavioral and non-behavioral features are both needed to model item attractiveness successfully, while they are fundamentally different. Non-behavioral features reflect intrinsic properties in the relationship between queries and items. They carry higher certainty than behavioral data and are well-defined for all queries and items.[1] In this sense, they are the "typical" features in a supervised machine learning problem where the task is to generalize. Behavioral features, on the other hand, are not defined (missing not at random) for items the system did not show. In contrast to non-behavioral features, they involve higher uncertainty and they are dynamic, as they are constantly changing with customer-system interactions. If all items were exposed sufficiently often to customers in response to a query, an optimal ranking could be produced based solely on a click rate estimated from behavioral features. However, this approach is not able to properly rank for new queries or new items with empty behavioral features. Behavioral features can therefore be seen as involving a memorization mechanism.

Despite this important difference, behavioral and non-behavioral features are commonly used in the same way in a traditional formulation of the ranking task as a supervised machine learning problem. We envision a better approach where we train an additional model based on non-behavioral features to impute or predict the behavioral feature values. The challenge here is how to predict behavioral features accurately with a non-behavioral feature model, given the lack of generalization power of the latter regarding item popularity, as we are precisely arguing here. We refer to this problem as "Learning and Memorizing to Rank" (LMTR), described in Section 3.4.

3.3 Related formulations

Aside from LTR frameworks that consider non-behavioral features and relevance only, some frameworks and algorithms address the combination of both types of information (the generalization and memorization issue) in a similar direction as we discussed above. We briefly summarize such work here and point out the differences to our proposed perspective.

In this area, LTR has been formulated as a multi-armed bandit problem under different click models, such as cascading bandit [22], DCM bandit [21] and position-based model bandit [23]. In such work, the goal is to find the optimal ranking that minimizes the cumulative regret (where click is the reward) for a single query. This formulation takes thus a pure memorization angle. A natural step forward is to model the LTR problem as a contextual bandit [26, 27], where the mean reward is modeled as a function of the context features, rather than a pure memorization of the reward distribution, in the hope to bring more efficiency in the bandit algorithms.

One difference between the usual contextual bandit framework and our perspective is that we model the mean reward (attractiveness of items given queries) as depending both on behavioral and non-behavioral features, whereas reward is modeled using only non-behavioral features in the contextual bandit. The source of uncertainty is therefore fundamentally different in the two problems: the non-behavioral feature model parameter uncertainty in the contextual bandit vs. the behavioral data uncertainty in LMTR. Since the key to the design of a bandit algorithm is reward uncertainty quantification to balance the exploration and exploitation, the nice empirical and theoretical results in existing contextual bandit LTR frameworks do not hold in LMTR.

Another closely related problem considered in the LTR literature is the combination of generalization and specialization. Jagerman et al. [18] and Oosterhuis and De Rijke [33] propose to train a general ranking policy that, in our terminology, is based on non-behavioral features only, along with different behavioral-only policies specialized for each query. A high confidence bound is then computed on the performance difference, based on a large amount of interaction data, and the specialized policy for a query is only deployed when it outperforms the general policy with sufficient confidence. Similar to what we discussed above, this clearly differentiates the

[1] By "certainty" in this context, we mean certainty in the system knowledge about the property the feature represents. For instance, the technical description of a new TV set is a known piece of information, whereas how successful the TV will be among consumers is an unknown variable that we can only speculate about, and estimate with increasing certainty as purchase data starts to become available.

role of the non-behavioral and behavioral features. However, behavioral features are used in this setting as a way to "fine tune" the general policies. In our LMTR perspective, we almost always need behavioral features, even in the general policy. Due to the lack of generalization power of non-behavioral features in predicting item popularity, we need a more fundamental and efficient use of behavioral features than just fine tuning.

3.4 Learning and memorizing to rank

We now formally define the LMTR setting. At time t, the user issues a query q_t, the system presents the match set D_{q_t} with ranking π_t powered by a ranker $f_t = f_t(q, d)$ and receives the reward as clicks $\boldsymbol{c}_t$. For a randomly selected query-item pair (q, d) at position k with non-behavioral feature X^{nb}, we model a click random variable C as a consequence of examination E and item attractiveness A. Following the notation in [44] that considers both position bias and trust bias, we have this affine relationship between click probability and the probability that the item is attractive given the query:

$$\mathbb{P}(C = 1|d, k, q) = \alpha_k \, \mathbb{P}(A = 1|d, q) + \beta_k$$

where $\alpha_k = \mathbb{P}(E = 1|k)(\mathbb{P}(C = 1|k, A = 1, E = 1) - \mathbb{P}(C = 1|k, A = 0, E = 1))$ and $\beta_k = \mathbb{P}(E = 1|k)\mathbb{P}(C = 1|k, A = 0, E = 1)$.

The estimation of the bias parameters α_k, β_k and the way of debiasing during model training are well studied [4, 5] and are not the focus of this paper. We consider the situation when the attractiveness A cannot be well captured by the non-behavioral features X^{nb} only, as discussed earlier. Formally, the best theoretical prediction of A based on X^{nb} is the conditional expectation $\mathbb{E}[A|X^{nb}]$ and we have the variance decomposition:

$$\mathrm{Var}(A) = \mathrm{Var}(\mathbb{E}[A|X^{nb}]) + \mathbb{E}[\mathrm{Var}(A|X^{nb})].$$

We define the prediction (generalization) power as:

$$\rho = \mathrm{Var}(\mathbb{E}[A|X^{nb}])/\mathrm{Var}(A) \tag{1}$$

which is a value between 0 and 1: when A and X^{nb} are independent we get $\rho = 0$ since $\mathbb{E}[A|X^{nb}] = \mathbb{E}[A]$ is a constant with 0 variance. When A fully depends on X^{nb}, that is, there is a function f such that $A = f(X^{nb})$, hence $\mathbb{E}[A|X^{nb}] = A$, and the ratio becomes $\rho = 1$. In general, we consider the system when ρ is small.

Since the prediction power of non-behavioral features is quite limited in our setting, we require the ranker to use both behavioral features $\boldsymbol{x}^b_t(q, d)$ and non-behavioral features $\boldsymbol{x}^{nb}(q, d)$. The former are calculated from historical data $\mathcal{D} = \{(q_i, D_{q_i}, \boldsymbol{c}_i)\}_{i=1}^n$ and are updated as new data comes: $\boldsymbol{x}^b_{t+1} = \phi(\boldsymbol{x}^b_t, q_{t+1}, D_{q_{t+1}}, \boldsymbol{c}_{t+1})$.

In practice, the ranker can be expected to require less frequent updates than features do. We can express the ranker parameters as $\boldsymbol{\theta}_t = (\boldsymbol{\theta}^b_t, \boldsymbol{\theta}^{nb}_t)$ which will also be learned from historical data and potentially be updated by new interaction data. The parameters in the ranker indicate the relative importance of features for optimizing the reward, and can be expected to be quite stable, unless there is a salient distribution change in the queries users issue or the available items in the system.

Hence for simplicity, we fix the ranker parameters and assume the ranking updates only come from behavioral feature updates: $f_t(q, d) = f(\boldsymbol{x}^b_t(q, d), \boldsymbol{x}^{nb}(q, d); \boldsymbol{\theta})$. Given the historical data $\mathcal{D}$, the goal is to learn the ranker parameters $\boldsymbol{\theta}$ and design behavioral features $\boldsymbol{x}^b_t$ such that ranking by $f_t = f(\boldsymbol{x}^b_t, \boldsymbol{x}^{nb}; \boldsymbol{\theta})$ maximizes the expected cumulative reward in a period of time T: $\sum_{t=0}^{T} \sum_{d \in D_{q_t}} \mathbb{E}c_{\pi_t(d)}$. Essentially, the learning component comes from estimating the ranker parameters θ from data, while the memorizing component depends on the behavioral signals captured in $\boldsymbol{x}^b$.

Later in section 5 we simulate a simple LMTR system and demonstrate that a ranker without feature exploration suffers from a severe cold-start effect when the prediction power ρ is low.

4 OVERCOMING COLD START THROUGH EMPIRICAL BAYES

We now present our approach to address cold start in e-commerce search based on the framework of LTMR described in Section 3. In a LMTR system, the ranking heavily relies on behavioral features that are often missing or involve high uncertainty for new query-item pairs. Such items get ranked lower than they should and it is hard for users to discover new high-value choices that might greatly match their needs. A typical discovery journey of a new item is first through spearfishing queries where the match set is of a much smaller size than average. The new item then gets a chance to be found by and exposed to users, and behavioral features can get an initial critical mass, just enough to trigger discoverability.

Once the behavioral features of the new items reach stability, they start to emerge also in more general queries. Therefore, it takes a very long time for new items to be stably ranked at their due position. The system needs an efficient and reliable mechanism to actively explore new products to accumulate behavioral information faster, in order to: i) help customers to easily discover relevant new products; and ii) help sellers to present their new selections in front of consumers.

Since the cold start problem in LMTR is mainly due to the uncertainty of behavioral features, which are the main factors of short-term ranking changes, it is natural to design the exploration mechanism through behavior prediction and update at query time when new interaction data comes in. We refer to this as online feature exploration. Two challenges for online feature exploration are:

(1) The prior models for behavioral features need to be efficient due to tight latency requirement.
(2) Once the explored items get customer feedback, the feature values need to be properly updated to reflect their preference.

4.1 Modeling engagement and update via Empirical Bayes

Conceptually, the Empirical Bayes (EB) approach [8] is well suited to the challenge: we can learn a simple model that determines an informative prior of behavioral features for each cold-start query-item pair based on their non-behavioral features to break the cold start, and then use Bayes' formula to update the feature value once we observe new feedback. This mechanism can shift promoted cold items towards the level of exposure they actually warrant.

In more detail, let us assume the behavioral feature is just the estimated click rate for a query-item pair: $\hat{p} = m/n$ where n is the number of times an item d has been impressed in response to a query q, and m is the number of times users clicked on the item in the impressed results. If a new product has never been returned for a query, or has only been showed few times, feature $\hat{p}$ is undefined or highly unreliable. To overcome this cold-start problem,

we assume the number of clicks m is drawn from a Binomial distribution $m|n,p \sim \text{Bin}(n,p)$, where the prior distribution of the true click rate p follows a Beta distribution $p \sim B(\alpha,\beta)$. We make this an informative prior by assuming the parameters α, β depend on the non-behavioral query-item feature vector $\boldsymbol{x}^{nb}(d,q)$ through a parameterized function $\alpha = g_1(\boldsymbol{x}^{nb},\theta_1)$ and $\beta = g_2(\boldsymbol{x}^{nb},\theta_2)$.

The parameters θ_1,θ_2 can be learned through maximum likelihood estimation: assuming we observe N query item pairs with the triplet $(\boldsymbol{x}_i^{nb}, n_i, m_i), i = 1,2,...,N$, the parameters are determined by maximizing the log-likelihood function of the observations:

$$\mathcal{L}(\theta_1,\theta_2) = \sum_i (\log B(\alpha_i + m_i, n_i + \beta_i - m_i) - \log B(\alpha_i,\beta_i))$$

where $\alpha_i = g_1(\boldsymbol{x}_i^{nb},\theta_1)$, $\beta_i = g_2(\boldsymbol{x}_i^{nb},\theta_2)$, and B is the Beta function. In the experiments we report later, we test both linear and tree models for g_1 and g_2. For linear models, the parameters can be solved by gradient descent. The tree models are learned in a usual greedy approach for regression trees, but adding a linear model on each leave, where the parameters are learned by gradient descent.

Once we have the informative priors, following Bayes' rule and the property of conjugate priors, we know that the posterior distribution of the click rate after observing the behavioral data (n,m) is still a Beta distribution: $p|m,n \sim B(\alpha+m, \beta+n-m)$. Then, to update the behavioral feature p after initialization with the learned prior model, we can simply update the behavioral feature to the posterior mean $\hat{p}_{eb} = (\alpha+m)/(\alpha+\beta+n)$.

4.2 Online feature exploration

EB-based feature exploration can be performed in two settings: offline and online. In the offline exploration approach, the feature value is updated based on logged customer feedback data, and published into the search engine through a common indexing pipeline. Since the calculations are done offline, we can afford much more complex prior models and update rules. However, because new products lack sufficient behavioral data, generating related queries for new products based on non-behavioral information is notoriously hard [14]. Moreover, the feature update latency is considerably long in the offline pipeline. In contrast, generating related queries for new products is not a problem in online exploration: we can explore all qualified new products recalled by the search engine at query time. Also importantly, online exploration avoids the long involved latency in a full offline loop. Consequently, we implement online feature exploration in our experiments.

In the online feature exploration framework (Figure 1), we use prior models to predict behavioral features of new products as an initial value. The prior models are hosted in the search engine and invoked at query time. We also build a fast customer feedback pipeline to continuously update customer feedback data, and publish it into the search engine as the input data for the behavioral feature update rules (based on Empirical Bayes as we describe in the next section). As we discuss later in our reported experiments in Sections 6 and 7, the update speed of the customer feedback pipeline is critical to the practical effectiveness of product exploration.

5 EXPERIMENTS ON A SIMULATED SYSTEM

We now empirically study the effect of feature exploration based on Empirical Bayes for mitigating cold-start. We first test our approach in a simulated LMTR system; later in Sections 6 and 7.4 we report

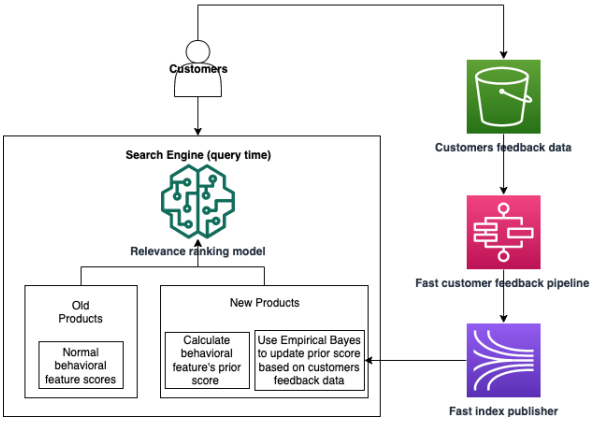

Figure 1: Online feature exploration with fast customer feedback pipeline.

experiments in a real, large-scale e-commerce website. In all experiments, we track the reward (clicks for the simulation, purchases for the real system) from all items and cold items over a period of time. We then compare the performance of the ranking with and without feature exploration, taking the former as a baseline.

Simulation-based experiments are common in the learning to rank and bias mitigation literature [2, 20, 32]. Typically, real data involving queries, documents, content features and relevance labels from expert annotation are taken from public datasets, and user clicks are simulated upon the real data based on a configured click model. In such settings, ranking models trained on click data can be compared to an oracle model trained on the expert annotation.

Our experiments are based on full simulation of content features and the true labels, rather than semi-simulated experiments upon expert annotations, for the following reasons. Simulations in prior work are based on public learning to rank datasets collected in the context of web search, which is quite different from e-commerce [9, 11, 35]. Expert annotation assumes a fair level of objectivity in topical relevance, which can go a long way in domains such as general web search. In e-commerce, the motivation for a customer to purchase a search result certainly involves a topical relevance requirement. But the simplifying assumption that any topically relevant product is as good as any other does not work when the choice among relevant results is the core of the problem. Key subjective factors beyond topicality are involved in making a product attractive given a query, such as price, convenience, fancy, and other intangible motivations, that cannot be captured by external judges.

As discussed in previous sections, content-based features have limited ability to predict attractiveness. With full simulation, where the true labels and non-behavioral features are also generated by us, it is easier to simulate different levels of prediction power of non-behavioral features. The purpose of this simulation is not to mimic the real distribution of customer feedback and features. Rather, we aim to show here that cold start becomes more severe when the prediction power of non-behavioral features is lower, and how the feature exploration can help to mitigate this. We describe the details of our LMTR system simulation in the subsections that follow.

5.1 Non-behavioral features

We generate 10,000 items and 1,000 queries described by feature vectors $\boldsymbol{x}_i^I$ and $\boldsymbol{x}_j^Q$ respectively. Then for each query q_j, we randomly

select a subset of items D_{q_j} as the match set, ranging from 5 to 50 items, with average size to be 25. A small match set indicates that the corresponding query is a spearfishing query. This way we get around 25,000 valid query-item pairs. We also generate a feature vector $\boldsymbol{x}_{ji}^{IQ}$, $j = 1, ..., 1{,}000$, $i = 1, ..., |D_{q_j}|$ for each pair. To simplify the simulation, we make all the feature vectors take size 1, and we assign their value uniformly at random in $[0, 1]$. The non-behavioral features we get are thus a length-3 vector $\boldsymbol{x}_{ji}^{nb} = (x_i^I, x_j^Q, x_{ji}^{IQ})$.

5.2 Product attractiveness

We model the mean attractiveness of a pair as $p_{ji} = w\, f(\boldsymbol{x}_{ji}^{nb}) + (1 - w)\, e_{ji}$, where f is a linear function with unit length coefficient vector, and e_{ji} is a latent random variable indicating the popularity of product d_i for query q_j, independent from the feature $\boldsymbol{x}_{ji}^{nb}$ and uniform in $[0, 1]$. The weight $w \in (0, 1)$ controls the prediction power of non-behavioral features for attractiveness. Given the feature, we simulate the attractiveness of a query-item pair by a binary random variable: $A_{ji}|\boldsymbol{x}_{ji}^{nb} \sim \text{Bern}(p_{ji})$. The prediction power of non-behavioral features defined earlier in Equation 1 then becomes:

$$\rho = \frac{\text{Var}[\mathbb{E}(A|X^{nb})]}{\text{Var}[A]} = \frac{w^2/12}{1/4} = \frac{w^2}{3}$$

where the final result follows from the definitions of A, p, f, and the mean and variance of Bernoulli and uniform distributions.

5.3 Interaction data and ranker training

We also simulate the interaction data collection, which we represent as a set of tuples $(q_j, d_i, n_{ji}, m_{ji})$, where n_{ji} is the number of impressions (number of times shown to users) of item d_i in response to query q_j, and $m_{ji} \le n_{ji}$ is the number of clicks produced on these impressions. We generate n_{ji} uniformly at random between 10 and 1,000 for all pairs, and the number of clicks from a Binomial distribution $m_{ji}|n_{ji} \sim Bin(p_{ji}, n_{ji})$, based on the mean attractiveness p_{ji} defined earlier. Note that since click debiasing is not the focus of this experiment, we assume the behavioral data we collect are free of position bias for simplicity. Finally, we randomly sample 1,000 items to simulate a cold-start item set, by removing all the behavioral data of the impression and click pairs involving those items. This set will be used as training data to train rankers. We also generate an independent copy of the data collection (the cold-start item set is the same) for the calculation of the behavioral feature (click rate $\hat{p}$).

We train two ranker types on the simulated interaction history:

- Non-behavioral ranker: the scoring function f^{nb} is trained by logistic regression taking only non-behavioral features $\boldsymbol{x}^{nb}$ as input to estimate the probability whether there will be a click.
- Behavioral ranker: similarly consists of a logistic regression scoring function f^b, but in addition to non-behavioral features, the function takes the estimated click rate $\hat{p}$ as input feature which is computed from historical interaction data.

The interaction data that the rankers are trained upon is independent from the historical interaction data for behavioral feature calculation, but they are generated through the same process.

5.4 Feedback loop simulation

Finally, we are ready to simulate the online interaction between users and the LMTR system. At each time t, we randomly sample a query q_t and obtain the match set D_{q_t}. We use the scoring function f to rank the items in decreasing order, and we simulate clicks on the top 10 items by the probability of attractiveness. This top-k selection bias, together with the uncertainty of behavioral features, form the cold-start challenges in LMTR.

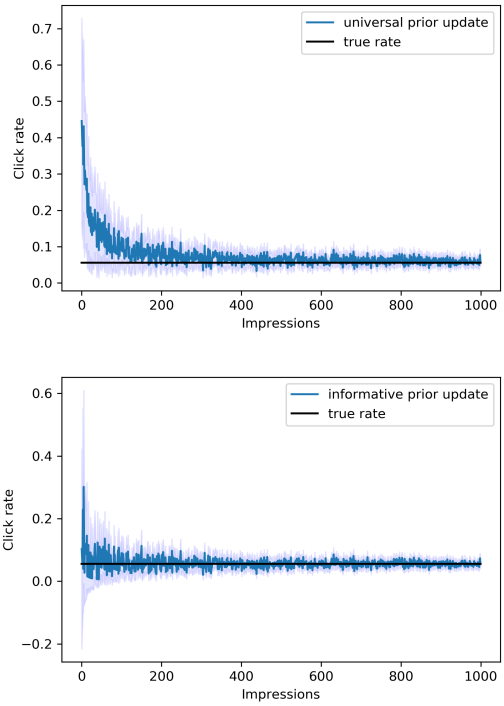

Figure 2: Posterior mean and variance estimate from informative vs. non-informative priors.

We consider three combinations of ranker and feature updating strategy: 1) only using a non-behavioral ranker $f^{nb} = f^{nb}(\boldsymbol{x}^{nb})$ without any feature updates; 2) using a behavioral ranker $f_t^b = f_t^b(\boldsymbol{x}^{nb}, \hat{p}_t)$ and updating the behavioral feature (the estimated click ratio $\hat{p}_t$) after each turn; 3) the same as 2, but the behavioral feature calculations and updates for the cold-start item set is carried out by Empirical Bayes $\hat{p}_{eb}$ as described in section 4.1.

First we calculate the informative prior Beta distribution where the parameters α, β depend on the non-behavioral features through a linear model. For comparison, we also compute the universal prior parameters which are independent of non-behavioral features. Both are calculated through maximum likelihood estimation.

5.5 Results

First, we examine the efficiency gain by using an informative prior instead of a universal prior. We see in Figure 2 that, as user feedback grows, the posterior estimation from the informative prior (the lower graph) and the universal prior (the upper graph) both converge to the true click rate. But the informative prior needs much fewer impressions to reach a stable and accurate click rate estimate since its initial guess is much closer to the true click rate.

Next, we analyze the cumulative reward over cold items and all items for the three ranking and updating methods: i) the non-behavioral ranker ii) the behavioral ranker and iii) the behavioral ranker with EB feature exploration (simply refer as EB). In Figure

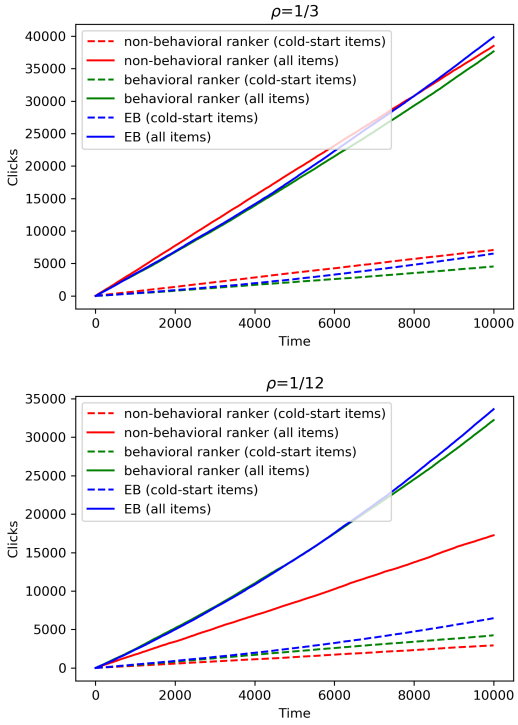

Figure 3: Reward over time for cold-start items and all items under three rankers.

3, the cumulative reward for the cold-start items and all items is plotted with dashed and solid line respectively. We also have two situations here: i) when the generalization power of the non-behavioral feature is strong (the upper graph) and ii) when the generalization power of the non-behavioral feature is weak (the lower graph). We can see in the figure that:

(1) When the prediction power is strong, the non-behavioral ranker is not bad in terms of accumulated reward, while the behavioral ranker is much better when the prediction power is low.
(2) When the prediction power is low, the cold-start problem is more severe: the average reward of the cold-start items is significantly lower.
(3) In terms of the cumulative reward from cold items, the behavioral ranker with feature exploration (EB) is much better than the non-behavioral ranker and the behavioral ranker without feature exploration. This supports our hypothesis that the main cause of the cold-start issue in a LMTR system is the lack of prediction power of non-behavioral features and the bias introduced by behavioral features. Feature exploration via EB shows promise here in mitigating the problem.

6 ONLINE A/B TEST

We now test our approach on real search traffic. In order to verify our hypothesis that online feature exploration can effectively mitigate cold-start, we first run a proof-of-concept experiment (EB v1) for our EB-based feature exploration, through an A/B test on 8 million queries during a week. In this experiment, we predict the prior value of the behavioral feature as discussed in the previous section, using a linear model, and we adjust the posterior using Empirical Bayes based on customer feedback.

Table 2: Online results of A/B test comparing Empirical Bayes with control search system. The statistical significance is denoted by * (p-value < 0.05).

	New products			All products
	Impression	Click	Purchase	Purchase
EB v1	+97.42%*	+57.99%*	+38.84%*	-0.5%*
EB v2	+13.53%*	+11.38%*	+11.14%*	+0.08%

It can be noted in the results reported in Table 2 (first row: EB v1) that our method significantly increases the impressions of new products (up by 97.42%) compared to the baseline. The improvement in clicks and purchases highlights the effectiveness of our method in overcoming cold start. This shows that our EB approach is quite effective in enhancing the exploratory behavior of our search system, giving fair opportunity to new products to be chosen by customers. This comes however at the expense of a statistically significant decrease in the overall purchase (including new and established products). Several factors may contribute to this outcome, in particular:

(1) *Old products have a higher purchase rate than new products*: in general, old products sell more due to the availability of rich metadata, customer reviews and other feedback. Therefore, showing new products and replacing old well-selling products may result in an overall lower short-term purchase rate.
(2) *Bayesian updates were not fast enough*: Bayesian updates with customer behavior are an opportunity to adjust the prior value and obtain a more informative posterior. In our experiments, we updated the posterior at a 24 hours interval. This means that suboptimal prior predictions remain exposed to customers for longer than they should, deteriorating the shopping experience and resulting in a lower purchase rate.

7 APPROACH REFINEMENTS

Based on the results and analysis of our first A/B test, we improve several components. Specifically, we improve the prior model, the update speed, and we introduce an early stopping strategy.

7.1 Improved prior model

Instead of a single linear model as in the previous experiment, we now train a tree-structured model with 8 leaves, where each leaf contains a linear model. The split features in the tree are pre-selected query-level features, such as query frequency; the split points and the coefficients of the linear models in the leaves are jointly trained by MLE as described in section 4.1.

We also incorporate behavioral features aggregated at product level in the model, instead of non-behavioral features only. The rationale for doing so is that these features should not introduce much bias against cold-start products as the training data only consists of cold-start query-product pairs, and the product-level behavioral features have good enough coverage among them (>90% non-zero values); while the query-product behavioral features have less than 10% non-zero values.

Product-level behavioral features are also quite powerful for identifying potentially popular cold products as they aggregate the customer behavior through spearfishing queries, and customer experiences outside of search. This makes product-level behavioral features a highly valuable source of prediction power even for cold items. Those changes resulted in about a 10% offline AUC increase.

7.2 Faster Bayesian update

The Bayesian update loop is the key component of the system. In this component, the update speed determines whether behavioral features can quickly converge to the right score: the faster the update, the better the system will perform. Keeping up with the massive volume of customer feedback data in an industrial search engine is a major challenge in this endeavor. We observed nonetheless that the volume of incremental data per hour is manageable, and this inspired us to split the stored feedback data into a "daily" set and an "hourly" set. The daily-updated data set spans the full lifecycle of products, whereas the hourly data set only stores data from engagement in the current day. Based on these two sets, we develop daily and hourly feedback pipelines. The daily pipeline is slow since it accumulates actions of all products. The hourly pipeline is fast and is published into the search engine through a real-time indexing system, that significantly improves the Bayesian update frequency from once every 24 hours to once every 2 hours.

7.3 Early stopping

Even with faster update, exploration incurs in considerable engagement loss when the prior model wrongly assigns a high behavioral feature value to a low-quality cold query-product pair. The ranking position of the explored item is boosted high for the query, yet the action rates (e.g. click, add-to-cart, purchase) of the item are much lower compared to other items in a similar position. Bayesian updates from the customer feedback will gradually correct the behavioral features, eventually ranking the item at a lower position that better corresponds to its inferior performance. However, this can take days if the prior model is too confident about the query-product pair, and the incurred loss can be substantial.

We directly mitigate this cost by implementing a UCB-based early stopping rule upon customer feedback. The rule is simple: we stop exploration for a query-product pair if the upper confidence bound of the dynamic user action rate is lower than a certain threshold τ. More specifically, based on a normal approximation of a Binomial distribution for m observed user actions over $n > 0$ impressions of an item, we stop exploration when:

$$\frac{m}{n} + z_a \sqrt{\frac{\tau(1-\tau)}{n}} < \tau.$$

where z_a is the $1 - a$ upper quantile of a standard normal distribution, $a \in [0, 1]$ being the confidence level of UCB. We set $a = 0.1$ in our implementation, thus getting $z_a = 1.6$ and a 90% upper confidence bound. In our experiments, we consider click, add-to-cart and purchase as actions and define the threshold τ as the average action rate of all non-cold-start pairs.

7.4 Second A/B test

We incorporated those improvements and run a second experiment (EB v2) through an A/B test on around 50 million queries, conducted during four weeks. The results are presented in Table 2 (second row: EB v2). The exploration is of much better efficiency than it was in the previous version. Specifically, we see that the increase in impression (13.53%), click (11.38%) and purchase (11.14%) is much better balanced – unlike the previous version, where engagement is much lower within the explored items. We even managed to increase the overall purchase by 0.08%. In addition to the above online metrics, we measured a small but statistically significant increase of 0.016% in nDCG@16. To compute nDCG, we pooled relevance judgments on a sample of 100,000 queries, for which the top 16 results were judged by majority vote among three human judges per query.

8 CONCLUSIONS

We find that cold start is still to much extent an unsolved problem in large-scale e-commerce services and highly competitive online markets. Cold start has been extensively addressed in recommender systems research, but we struggle to find comparable references in search-oriented retrieval, and e-commerce in particular where, as in recommender systems, user engagement is a fundamental signal to optimize user satisfaction. Cold start can be characterized as a case of selection bias, linking to extensive recent efforts on bias mitigation in learning to rank. Fully missing data is however a drastic case of bias for which available solutions tend to fall short.

As a direction to frame and address the problem, we cast search cold start as a Learning and Memorizing to Rank problem, where the key characteristic is that both non-behavioral and behavioral features are used as input for generating rankings. While real-world ranking systems are routinely using behavioral features [15, 41], work in the learning to rank literature addressing this perspective in search systems seems scant. Behavioral features are crucial in a LMTR system, but treating them naively as regular content-based features in a supervised learning strategy may result in a severe bias against items that lack sufficient feedback – we believe this effect may be even stronger than the bias in training labels. We thus envision the smart combination of the generalization power of non-behavioral features and the memorization power of behavioral features as the key to enable effective rankings in a LMTR system.

We provide a principled and scalable solution through behavioral feature exploration based on Empirical Bayes, where the problem is decomposed into an effective prediction of prior engagement probability, followed by suitable feature update, and exploration stopping mechanisms once engagement catches on. Our experiments with both a simulated and a real ranking system show highly positive results. Even if a moderate initial tradeoff in short-term engagement might be acceptable, we achieve cold-start remediation within bounds of unaffected business metrics. We also find that specific technical enhancements and adjustments, such as the accuracy of modular predictors and exploration update speed, are key to make the most of our proposed approach.

Beyond our reported efforts, the problem remains open and many directions unfold for continued research. Further ambitions would aim, for instance, to monitor a longer-term scale, beyond the short span of an A/B test. This might uncover further room for leveraging opportunity from new products, and justify more aggressive short-term tradeoffs to move beyond local minima. In this perspective, quantifying long-term improvements, and attributing them to specific treatments is a well-known challenge [15]. We also envision the formal definition of a LMTR bandit system as a direction worthy being explored, based on e.g. lower-bound regret analysis and the construction of algorithms that match the bounds. Closely related to the cold-start problem, fairness and diversity in LMTR are also important angles that should be addressed. A difficult challenge in such directions is properly measuring the improvements, defining the appropriate metrics that capture the intended effects, or even defining what those effects should be at the conceptual level.

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering* 17, 6 (June 2005), 734–749. https://doi.org/10.1109/TKDE.2005.99

[2] Aman Agarwal, Kenta Takatsu, Ivan Zaitsev, and Thorsten Joachims. 2019. A general framework for counterfactual learning-to-rank. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2019)*. 5–14.

[3] Aman Agarwal, Xuanhui Wang, Cheng Li, Michael Bendersky, and Marc Najork. 2019. Addressing Trust Bias for Unbiased Learning-to-Rank. (2019), 4–14. https://doi.org/10.1145/3308558.3313697

[4] Aman Agarwal, Ivan Zaitsev, and Thorsten Joachims. 2018. Counterfactual learning-to-rank for additive metrics and deep models. *arXiv preprint arXiv:1805.00065* (2018).

[5] Aman Agarwal, Ivan Zaitsev, Xuanhui Wang, Cheng Li, Marc Najork, and Thorsten Joachims. 2019. Estimating position bias without intrusive interventions. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining (WSDM 2019)*. 474–482.

[6] Iman Barjasteh, Rana Forsati, Farzan Masrour, Abdol-Hossein Esfahanian, and Hayder Radha. 2015. Cold-Start Item and User Recommendation with Decoupled Completion and Transduction. In *Proceedings of the 9th ACM Conference on Recommender Systems* (Vienna, Austria) *(RecSys 2015)*. ACM, New York, NY, USA, 91–98. https://doi.org/10.1145/2792838.2800196

[7] Christopher J.C. Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. (2010). https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview

[8] George Casella. 1985. An Introduction to Empirical Bayes Data Analysis. *The American Statistician* 39, 2 (1985), 83–87.

[9] Olivier Chapelle and Yi Chang. 2011. Yahoo! learning to rank challenge overview. In *Proceedings of the learning to rank challenge*. PMLR, 1–24.

[10] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. 2015. Click models for web search. *Synthesis lectures on information concepts, retrieval, and services* 7, 3 (2015), 1–115.

[11] Domenico Dato, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Nicola Tonellotto, and Rossano Venturini. 2016. Fast ranking with additive ensembles of oblivious and non-oblivious regression trees. *ACM Transactions on Information Systems* 35, 2 (2016), 1–31.

[12] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle, and Lars Schmidt-Thieme. 2010. Learning Attribute-to-Feature Mappings for Cold-Start Recommendations. In *Proceedings of the 10th IEEE International Conference on Data Mining* (Sydney, Australia) *(ICDM 2010)*. IEEE Computer Society, 176–185. https://doi.org/10.1109/ICDM.2010.129

[13] Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamanic, Chen Wu, W. BruceCroft, and Xueqi Cheng. 2020. A Deep Look into neural ranking models for information retrieval. *Information Processing & Management* 57, 6 (2020). https://doi.org/10.1016/j.ipm.2019.102067

[14] Parth Gupta, Tommaso Dreossi, Jan Bakus, Yu-Hsiang Lin, and Vamsi Salaka. 2020. Treating Cold Start in Product Search by Priors. In *Companion Proceedings of the Web Conference 2020*. ACM, New York, NY, USA, 77–78. https://doi.org/10.1145/3366424.3382705

[15] Malay Haldar, Prashant Ramanathan, Tyler Sax, Mustafa Abdool, Lanbo Zhang, Aamir Mansawala, Shulin Yang, Bradley Turnbull, and Junshuo Liao. 2020. Improving Deep Learning for Airbnb Search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2020)*. ACM, New York, NY, USA, 2822–2830.

[16] Katja Hofmann, Anne Schuth, Shimon Whiteson, and Maarten De Rijke. 2013. Reusing historical interaction data for faster online learning to rank for IR. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining (WSDM 2013)*. 183–192.

[17] Ziniu Hu, Yang Wang, Qu Peng, and Hang Li. 2019. Unbiased lambdamart: an unbiased pairwise learning-to-rank algorithm. In *Proceedings of the World Wide Web Conference*. 2830–2836.

[18] Rolf Jagerman, Ilya Markov, and Maarten De Rijke. 2020. Safe exploration for optimizing contextual bandits. *ACM Transactions on Information Systems* 38, 3 (2020), 1–23.

[19] Shan Jiang, Yuening Hu, Changsung Kang, Tim Daly, Dawei Yin, Yi Chang, and Chengxiang Zhai. 2016. Learning Query and Document Relevance from a Web-Scale Click Graph. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Pisa, Italy) *(SIGIR 2016)*. ACM, New York, NY, USA, 185–194. https://doi.org/10.1145/2911451.2911531

[20] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased Learning-to-Rank with Biased Feedback. (2017), 781–789. https://doi.org/10.1145/3018661.3018699

[21] Sumeet Katariya, Branislav Kveton, Csaba Szepesvari, and Zheng Wen. 2016. DCM bandits: Learning to rank with multiple clicks. In *Proceedings of the 33rd International Conference on Machine Learning (ICML 2016)*. PMLR, 1215–1224.

[22] Branislav Kveton, Csaba Szepesvari, Zheng Wen, and Azin Ashkan. 2015. Cascading bandits: Learning to rank in the cascade model. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*. PMLR, 767–776.

[23] Paul Lagrée, Claire Vernade, and Olivier Cappe. 2016. Multiple-play bandits in the position-based model. *Advances in Neural Information Processing Systems* 29 (2016).

[24] Damien Lefortier, Pavel Serdyukov, and Maarten De Rijke. 2014. Online exploration for detecting shifts in fresh intent. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM 2014)*. 589–598.

[25] Jingjing Li, Mengmeng Jing, Ke Lu, Lei Zhu, Yang Yang, and Zi Huang. 2019. From Zero-Shot Learning to Cold-Start Recommendation. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence and 31st Innovative Applications of Artificial Intelligence Conference and 9th AAAI Symposium on Educational Advances in Artificial Intelligence* (Honolulu, HI, USA) *(AAAI/IAAI/EAAI 2019)*. AAAI Press, Article 514. https://doi.org/10.1609/aaai.v33i01.33014189

[26] Shuai Li, Tor Lattimore, and Csaba Szepesvári. 2019. Online learning to rank with features. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*. PMLR, 3856–3865.

[27] Shuai Li, Baoxiang Wang, Shengyu Zhang, and Wei Chen. 2016. Contextual combinatorial cascading bandits. In *Proceedings of the 33rd International conference on machine learning (ICML 2016)*. PMLR, 1245–1253.

[28] Tie-Yan Liu. 2011. Learning to rank for information retrieval. (2011).

[29] Paul Missault, Arnaud de Myttenaere, Andreas Radler, and Pierre-Antoine Sondag. 2021. Addressing Cold Start With Dataset Transfer In E-Commerce Learning To Rank. In *The Web Conference 2021 Workshop on Knowledge Management in E-Commerce*. https://www.amazon.science/publications/addressing-cold-start-with-dataset-transfer-in-e-commerce-learning-to-rank

[30] Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep Content-Based Music Recommendation. In *Proceedings of the 26th International Conference on Neural Information Processing Systems* (Lake Tahoe, NV, USA) *(NIPS 2013)*. Curran Associates Inc., Red Hook, NY, USA, 2643–2651.

[31] Harrie Oosterhuis and Maarten de Rijke. 2018. Differentiable Unbiased Online Learning to Rank. (2018), 1293–1302. https://doi.org/10.1145/3269206.3271686

[32] Harrie Oosterhuis and Maarten de Rijke. 2021. Unifying online and counterfactual learning to rank: A novel counterfactual estimator that effectively utilizes online interventions. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining (WSDM 2021)*. 463–471.

[33] Harrie Oosterhuis and Maarten de de Rijke. 2021. Robust Generalization and Safe Query-Specializationin Counterfactual Learning to Rank. In *Proceedings of the Web Conference*. 158–170.

[34] Zohreh Ovaisi, Ragib Ahsan, Yifan Zhang, K. Vasilaky, and E. Zheleva. 2020. Correcting for Selection Bias in Learning-to-rank Systems. (2020), 1863–1873. https://doi.org/10.1145/3366423.3380255

[35] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 datasets. *arXiv preprint arXiv:1306.2597* (2013).

[36] Mark Sanderson. 2010. *Test collection based evaluation of information retrieval systems*. Now Publishers Inc.

[37] Martin Saveski and Amin Mantrach. 2014. Item Cold-Start Recommendations: Learning Local Collective Embeddings. In *Proceedings of the 8th ACM Conference on Recommender Systems* (Foster City, CA, USA) *(RecSys 2014)*. ACM, New York, NY, USA, 89–96. https://doi.org/10.1145/2645710.2645751

[38] Anne Schuth, Harrie Oosterhuis, Shimon Whiteson, and Maarten de Rijke. 2016. Multileave gradient descent for fast online learning to rank. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining (WSDM 2016)*. 457–466.

[39] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, Lexing Xie, and Darius Braziunas. 2017. Low-Rank Linear Cold-Start Recommendation from Social Data. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence* (San Francisco, CA, USA) *(AAAI 2017)*. AAAI Press, 1502–1508.

[40] Suvash Sedhain, Scott Sanner, Darius Braziunas, Lexing Xie, and Jordan Christensen. 2014. Social Collaborative Filtering for Cold-Start Recommendations. In *Proceedings of the 8th ACM Conference on Recommender Systems* (Foster City, CA, USA) *(RecSys 2014)*. ACM, New York, NY, USA, 345–348. https://doi.org/10.1145/2645710.2645772

[41] Daria Sorokina and Erick Cantu-Paz. 2016. Amazon Search: The Joy of Ranking Products. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Pisa, Italy) *(SIGIR 2016)*. ACM, New York, NY, USA, 459–460. https://doi.org/10.1145/2911451.2926725

[42] Sumit Taank, Tri Minh Cao, and Abhishek Gattani. 2017. Re-ranking results in a search. https://patents.google.com/patent/US9563705 Patent No. US9563705B2, Filed Mar. 20th, 2013, Issued Feb. 17th, 2017.

[43] Manos Tsagkias, Tracy Holloway King, Surya Kallumadi, Vanessa Murdock, and Maarten de Rijke. 2021. Challenges and Research Opportunities in ECommerce Search and Recommendations. *SIGIR Forum* 54, 1, Article 2 (Feb. 2021). https://doi.org/10.1145/3451964.3451966

[44] Ali Vardasbi, Harrie Oosterhuis, and Maarten de Rijke. 2020. When inverse propensity scoring does not work: Affine corrections for unbiased learning to rank. In *Proceedings of the 29th ACM International Conference on Information &*

Knowledge Management (CIKM 2020). 1475–1484.

[45] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. 2017. DropoutNet: Addressing Cold Start in Recommender Systems. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, CA, USA) *(NIPS 2017)*. Curran Associates Inc., Red Hook, NY, USA, 4964–4973.

[46] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. Learning to Rank with Selection Bias in Personal Search. (2016), 115–124. https://doi.org/10.1145/2911451.2911537

[47] Wei Wu, Hang Li, and Jun Xu. 2013. Learning Query and Document Similarities from Click-through Bipartite Graph with Metadata. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining* (Rome, Italy) *(WSDM 2013)*. ACM, New York, NY, USA, 687–696. https://doi.org/10.1145/2433396.2433481

[48] Jun Xu, Xiangnan He, and Hang Li. 2020. Deep Learning for Matching in Search and Recommendation. *Foundations and Trends in Information Retrieval* 14, 2-3 (2020), 102–288. https://doi.org/10.1561/1500000076

[49] Tao Yang, Chen Luo, Hanqing Lu amd Parth Gupta, Bing Yin, and Qingyao Ai. 2022. Can clicks be both labels and features? Unbiased Behavior Feature Collection and Uncertainty-aware Learning to Rank. (2022). https://doi.org/10.1145/2911451.2911537

[50] Yisong Yue and Thorsten Joachims. 2009. Interactively optimizing information retrieval systems as a dueling bandits problem. (2009), 1201–1208. https://doi.org/10.1145/1553374

[51] Ziwei Zhu, Shahin Sefati, Parsa Saadatpanah, and James Caverlee. 2020. Recommendation for New Users and New Items via Randomized Training and Mixture-of-Experts Transformation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2020)*. ACM, New York, NY, USA, 1121–1130.