Towards Optimization of E-Commerce Search and Discovery

Anjan Goswami
University of California, Davis
agoswami@ucdavis.edu

ChengXiang Zhai
University of Illinois
Urbana-Champaign, IL
czhai@illinois.edu

Prasant Mohapatra
University of California, Davis
pmohapatra@ucdavis.edu

ABSTRACT

E-Commerce (E-Com) search is an emerging problem with multiple new challenges. One of the primary challenges constitutes optimizing multiple objectives involving business metrics such as sales and revenue and maintaining a discovery strategy for the site. In this paper, we formalize the e-com search problem for optimizing metrics based on sales, revenue, and relevance. We define a notion of item discoverability in search and show that learning to rank (LTR) algorithms trained with behavioral features from e-com customer interactions (eg. clicks,cart-adds, orders etc.) do not by themselves address the discoverability problem. Instead, a suitable explore-exploit framework must be integrated with the ranking algorithm. We thus construct a practical discovery strategy by keeping a few top positions for discovery and populating some of the items selected through exploration. Then, we present a few exploration strategies with low regret bounds in terms of business metrics. We conduct a simulation study with a synthetically generated dataset that represents items with different utility distribution and compares these strategies using metrics based on sales, revenue, relevance, and discovery. We find that a strategy based on adaptive submodular function based discovery framework can provide a nice balance of business metrics and discoverability compared to other strategies based on random exploration or multi-armed bandit. However, another strategy, based on monotonic submodular optimization function that needs to be integrated with linear LTR models also works well for discovery and has nice performances with respect to sales, revenue, and relevance.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: E-Com Search

General Terms

Algorithms

Keywords

e-com search, retrieval models, learning to rank, discoverability, exploration-exploitation

1. INTRODUCTION

One of the most critical components of an e-commerce (e-com) marketplace is its search functionality. The goal of an e-commerce search engine is to help the buyers to find and purchase their desirable products. The buyers' intent represents the demand side and the products listed by the sellers represent the supply side of an e-com platform. The e-com search engine's function can be considered to provide an efficient matching platform for the supply and the demand side and to facilitate transactions to generate revenue. Like a web search engine, an e-com search also intends to ensure user satisfaction by presenting the most relevant items that a user is searching for. However, an e-com search engine generates revenue directly based on the products presented to the users; this is in sharp contrast with how a web search engine generates revenue through advertising and raises significant new challenges for designing effective search algorithms. Thus besides the usual goal of maximizing relevance of the search results, an e-com search engine also needs to optimize two additional goals that generally do not exist in web search scenarios, i.e., it has to 1) maximize the business metrics such as revenue or sales from the purchases and 2) minimize the inventory cost by selling the items faster. Both the objectives are dependent on the performance of the underlying ranking algorithm of the search engine other than the assortment selection and demand generation. The first set of goals of maximizing the business metrics can be achieved by constructing a rank function using learning to rank framework with a suitable optimization objective that is in line with the business goals of the e-commerce company. The second goal however requires the e-commerce sites to have a strategy of discovery so that it can expose as many items to the customers faster and determine what products are selling. This allows the companies to readjust their assortment strategy to optimize the inventory and associated costs. The discovery strategies however, can hurt the business goals since it requires showing previously unexplored items to the customers. In e-commerce, learning to rank functions are trained by constructing mainly two types of features: (a) features that arise from static contents of the product description (b) features that come from the behavioral signals such as clicks, cart-adds, sales, and review ratings. It has been observed [16, 21] that the second group of features is more useful for finding a comparative relevance

among the items. However, the use of the machine learning algorithms in search engines tends to bias a search engine toward favoring the viewed items by users due to the use of features that are computed based on user behavior such as clicks, rates of "add to cart", etc. Since a learning algorithm would rank items that have already attracted many clicks on the top, it might overfit to promote the items viewed by users. As a result, some items might never have an opportunity to be shown to a user (i.e., "discovered" by a user), thus also losing the opportunity to potentially gain clicks. Such "undiscovered" products would then have to stay in the inventory for a long time incurring extra cost and hurting satisfaction of the product providers. Hence, it is important to consider the integrating the aspect of discovery with the ranking mechanism for e-commerce search. In this paper, we provide a practical yet theoretically sound framework for learning to rank where it is possible to have a regulated discovery mechanism minimizing the loss in revenue, sales, or relevance metrics.

The key contributions of this paper can be described as follows:

(1) This is the first paper that formalizes an e-com search problem as a multi-objective LTR problem with continuous exploration that we call learning to rank and discover problem. (2) We discuss how we can construct a solution to this problem using the existing framework of multi-armed bandit and auction mechanism. (3) We also provide a novel paradigm combining algorithms from multi-objective optimization and explore and exploit paradigms.

2. BACKGROUND

2.1 LTR and Ecommerce Search

Learning to rank (LTR) algorithms [19] involve learning a function for optimizing a relevance measure using an offline labeled training data set. This function can be a neural net, boosted tree [3, 22], support vector machine [15] or linear models [13] etc. The common ranking measures [14] include normalized discounted cumulative gain (NDCG), mean reciprocal ranking (MRR), mean average precision (MAP) etc. The loss functions based on these ranking measures are mostly non-smooth, however researchers [5, 17] derive LTR algorithms such as LambdaMart [4] to address such problems. There is also a work on multi-objective learning to rank [22] that uses a click rate along with the human judged ratings to augment the computation of gradient direction from pairwise training data in LambdaMart algorithm. This algorithm can potentially be modified to learn to maximize multiple graded objectives where the priorities are known in advance. There are not too many papers on LTR algorithms applied to e-com data. Recently, researchers from Walmart [16] have conducted some experiments with multiple LTR algorithms with e-com data. In their observation, LambdaMart algorithm with a sales and click rate based objective worked well for their data set. In a separate paper, researchers from Amazon [21] reported use of gradient boosted tree based regression methods [8] for LTR application. They also reported using various normalization techniques to avoid the biases generated due to the heavy reliance of historical behavioral data in regression.

2.2 Online learning and Multi-armed bandit

Multi-armed bandit (MAB) algorithms [11, 20] have a rich literatures. The algorithms that are commonly used for exploration include ϵ-greedy [23], and variants of upper confidence bound (UCB) algorithm [2]. Additionally, k-armed dueling bandit framework [25] has been developed for finding out optimal bandit in a scenario where each bandit can be considered to optimize a specific objective. It is possible to frame a multiobjective problem with exploration component using such paradigm. In this paper, we aim to construct a framework that is simpler than K-armed bandit but more practical.

3. OPTIMIZATION OF E-COM SEARCH

We consider the problem of optimization of an E-Com search engine over a period of time $\{1, \cdots, T\}$. Let us assume that there are N queries, denoted by $Q = (q_1, q_2, \cdots, q_N)$, that can be sent to the search engine during this time. Let $\mathcal{Z} = \{\zeta_1, \cdots, \zeta_M\}$ be the set of M items and let the corresponding prices of the items be given by $\{\rho_1, \cdots, \rho_M\}$. Now, consider a rank function π and $\pi(q_i, t)$ outputs top K items which is a permutation from the set $\mathcal{Z}$. Now, let us consider two binary random variables $\Lambda^S_{ijt}, \Lambda^C_{ijt}$ where the first one denotes if a sale happens or not and the second one denotes if there is a click or not for an item ζ_j given a query q_t at time t. Now, given that we have these two binary random variables, we can define the probability of sales given a query q_i, item ζ_j at time t as $p(\Lambda^S_{ijt} = 1|q_i, \zeta_j, t)$ and similarly probability of click for the same can be defined as $p(\Lambda^C_{ijt} = 1|q_i, \zeta_j, t)$. Let's also create a normalization constant Γ that denotes the total number of search sessions during the time period we are considering to conduct this study.

Now, we can define several objective functions for the e-commerce search such as the following:

Conversion rate per visit (CRV):

$$g_{CRV} = \frac{1}{\Gamma} \sum_{t=1}^{t=T} \sum_{i=1}^{i=N} \sum_{j=1}^{j=M} p(\Lambda^S_{ijt} = 1|q_i, \zeta_j, t)$$

Revenue per visit (RPV):

$$g_{RPV} = \frac{1}{\Gamma} \sum_{t=1}^{t=T} \sum_{i=1}^{i=N} \sum_{j=1}^{j=M} p(\Lambda^S_{ijt} = 1|q_i, \zeta_j, t)\rho_j$$

We now also define a relevance based objective as follows:

$$g_{REL}(\pi) = \frac{1}{\Gamma} \sum_{t=1}^{t=T} \sum_{i=1}^{i=N} RM(\pi(q_i, t))$$

where RM can be any relevance measure such as normalized discounted cumulative gain, which is generally defined based on how well the ranked list $\pi(q)$ matches the ideal ranking produced based on human annotations of relevance of each item to the query. The aggregation function does not have to be a sum (over all the queries); it can also be, e.g., the minimum of relevance measure over all the queries, which would allow us to encode the preference for avoiding any query with a very low relevance measure score.

$$g_{REL}(\pi) = min_{i\in[1,N], j\in[1,T]} RM(\pi(q_i, t))$$

Now,we define a notion of discovery for an e-com engine.

To formalize the notion of discoverability, we say that the LTR function f is β-discoverable if all items are shown at least β times. Now, we can further define a β-discoverability rate as the percentage of items that are impressed at least

β times in a fixed period of time. Let us now define again a binary variable γ_i for every item ζ_i and then assume that $\gamma_i = 1$ if the item got shown in the search results for β times and $\gamma_i = 0$ in case the item is not shown in the search results more than β times. We can express this as follows:

$$g_{\beta-discoverability} = \frac{\sum_{i=1}^{i=|\mathcal{Z}|} \gamma_i}{|\mathcal{Z}|}$$

We can further define another objective sales through rate (STR) to denote the number of items sold in a period of time as percentage of the inventory. This objective represent the efficiency of selling items faster. Let us define again a binary variable λ_s for every item ζ_s and then assume that $\lambda_s = 1$ if the item got sold and $\lambda_s = 0$ in case the item is not sold.

Thus, the STR objective function when using policy π can be written as follows:

$$g_{STR}(\pi) = \sum_{i=1}^{i=|\mathcal{Z}|} \lambda_i$$

Now, we can define the objective of an e-com search engine can be the following:

$$\text{maximize} \quad g_{\text{CRV}}, g_{\text{RPV}}, g_{\text{REL}}, g_{\beta-\text{discoverability}}, g_{\text{STR}}$$

This is a multi-objective optimization and has an exploration component in the form of discovery. Furthermore, there can be business priorities that may need to in the consideration for formulating the problem.

4. STRATEGIES FOR SOLVING THE OPTIMIZATION PROBLEM

Since there are multiple objectives to optimize, it is impossible to directly apply an existing Learning to Rank (LTR) method to optimize all the objectives. However, there are multiple ways to extend an LTR method to solve the problem as we will discuss below.

4.1 Direct extension of LTR

One simple strategy is to somehow combine the multiple objectives with appropriate weighting parameters so as to form one single objective function, which can then be used in a traditional LTR framework to find a ranking that would optimize the consolidated objective function. The advantage of this approach is that we can directly build on the existing LTR framework, though the new objective function would pose new challenges in designing effective and efficient optimization algorithms to actually compute optimal rankings. One disadvantage of this strategy is that we cannot easily control the tradeoff between different objectives (e.g., we sometimes may want to set a lower bound on one objective rather than to maximize it). To address this limitation, we may allow some objectives to be treated as constraints.

4.2 Incremental Optimization of e-Com Search

An alternative strategy is to take an existing LTR ranking function as a basis and seek to improve the ranking (e.g., by perturbation) so as to optimize multiple objectives as described above; such an incremental optimization strategy is more tractable as we will be searching for solutions to the optimization problem in the neighborhood of an existing ranking function. Specifically, we instantiate the optimization problem discussed in the previous section by assuming that the engine incrementally perturbs a learning to rank function slightly in a way that the original relevance does not vary too much and then each function $\pi = (f_1, ..., f_T)$ at time step $t = 1$ to $t = T$ gradually improves upon multiple objectives. Intuitively, we start with a function f and generate the next function in the "neighborhood" of an existing ranking function f within a relevance bound and we take each step in a way that it attempts to improve the sales and revenue with high probability and it also tries to achieve better β-discoverability and STR for the items in its inventory. However, the search space for all such ranking functions is still intractable and we need a heuristic to obtain a "good" Pareto optimal solution for the multi objective optimization problem.

We thus need to construct an optimization framework that integrates a regulated exploration with an LTR paradigm.

4.3 Realization of multi-objective optimization with exploration

One way of constructing a framework that optimizes an e-com site for multiple business objectives along with an exploration option can be to use a fixed number of rank positions say x for exploration and use the rest of the top $(K - x)$ positions for exploitation. In this set-up, we can construct an explore set of items $\mathcal{E}$ where all the items have been shown less than β times. We then select top x candidates for a query q_i from the intersection of the recall set of items for the query and the set $\mathcal{E}$. Let's call this set corresponding to query q_i as $\mathcal{E}_i$. The items selected for the rest of the $(K - x)$ positions can be based on a suitable multi-objective LTR ranking function. This is a framework that is practical and the cost of the exploration can be estimated by the loss of revenue for selecting not the most optimal items for x positions. Note, it is also possible to select any rank positions to show the explored items. However, in this paper, let us assume that we keep those x items at top for keeping the framework simple.

4.4 LTR and Discoverability

It is straightforward to argue that a regular LTR function is not optimal for discoverability.

In an e-com site, only top K items are shown to the customers. Suppose there is an item ζ that is relevant to a set of queries Q. Now, assume that for each query $q \in Q$ there are K other relevant old items for which $f(\mathbf{X2})) > 0$. Assuming that the K items are equally relevant, we can say that the ζ has same value of $f(\mathbf{X1})$ compared to the k old items. Then, ζ will always rank below the top K items for the set of queries in Q. Suppose the current number of impressions for ζ is less than β, then it has no chance of receiving impression under the ranking framework using function f because it will always be ranked below the top K items unless there is new query that ranks this item higher. This means that as long as we can find a percentage of items that receives below β impressions, it is hard to improve the β-discoverability using a traditional LTR mechanism without any integrated discovery mechannism.

5. DISCOVERY STRATEGIES

In this section, we propose several strategies to select the x items for discovery that aims to minimize the loss of business metrics.

5.1 Auction based strategy

In this strategy, the items for top x positions can be assigned based on an auction strategy where sellers can bid for the positions. This strategy has been used in sponsored search for search engine companies such as Google and in sponsored product for e-commerce company such as Amazon. Design of such auction mechanism that balances the revenue and relevance is a complex subject and has been discussed in recent papers by Ghose et al. [10] and Athey at al. [1]. However, many e-commerce businesses may not want to use auction strategy to aid discovery since it can hurt the growth of seller participation on those sites.

5.2 Exploration with LTR (eLTR)

This is a strategy discussed in a recent paper by Goswami et al. [12]. In this strategy, we define the set from which the LTR function selects the items as $L_i \subset R_i$ for a given query q_i. We assume that all the items outside set L_i are not β-discoverable. Then, $L = \cup_{i=1}^{i=N} L_i$ is the set of all β-discoverable items. Hence, the set $E = R \setminus L$ can then be consisting of all the items that require exploration.

Goswami et al. [12] discuss three strategies to incorporate discovery in an e-commerce search.

Random selection based exploration from the recall set (RSE): This is a baseline strategy for continuous exploration with a LTR algorithm. In this, for every query q_i, we randomly select x items from the set $E \cap R_i$. Then, we put these x items on top of the other $(k-x)$ items that are selected using LTR from the set R_i. The regret here will be linear with the number of search sessions with exploration.

Upper confidence bound (UCB) based exploration from the recall set (UCBE): This is another simple strategy that uses a variant of UCB based algorithm for exploration instead of random sampling. Here, we maintain a MAB for each query. We consider each item in the set $E \cap R_i$ as an arm for the MAB corresponding to a query q_i. We maintain an UCB score for each of those items based on sales over impression for the query. If an item ζ_j is in the set $E \cap R_i$ and is shown b_j times in T iterations, and is sold a_j times in between, then the UCB score of the item ζ_j is $ucb_j = \frac{a_j}{b_j} + \sqrt{\frac{2 \log_2 T+1}{b_j}}$. Note, this is for a specific query. We then select x items based on top UCB scores.

Explore LTR (eLTR In this, we define a function that we call explore LTR (eLTR) to select the x items. The rest of the items for top K can be chosen using the traditional LTR. Then, we can either keep the x items on top or we can rerank all K items based on eLTR.

The main motivation for the eLTR is the observation that there is inherent overfitting in the regular ranking function used in an e-com search engine that hinders exploration, i.e., hinders improvement of β-discoverability and STR. The overfitting is mainly caused by a subset of features derived from user interaction data. Such features are very important as they help inferring a user's preferences, and the overfitting is actually desirable for many repeated queries and for items that have sustained interests t users (since they are "test cases" that have occurred in the training data), but it gives old items a biased advantage over the new items, limiting the increase of β-discoverability and STR. Thus the main idea behind e-LTR is thus to separate such features and introduce a parameter to restrict their influences on the ranking function, indirectly promoting STR. Formally, we note that in general, a ranking function can be written in the following form:

$$y = f(\mathbf{X}) = g(f_1\mathbf{X1}), f_2(\mathbf{X2}))$$

where $y \in \mathbb{R}$ denotes a ranking score and $\mathbf{X} \in \mathbb{R}^N$ is a N dimensional feature vector, $\mathbf{X1} \in \mathbb{R}^N 1$ and $\mathbf{X2} \in \mathbb{R}^N 2$ are two different groups of features such that $N1 + N2 = N, \mathbf{X1} \cup \mathbf{X2} = \mathbf{X}$. The two groups of features are meant to distinguish features that are unbiased (e.g., content matching features) from those that are inherently biased (e.g., clickthrough-based features). Here g is an aggregation function which is monotonic with respect to both arguments. It is easy to show that any linear model can be written as a monotonic aggregation function. It is not possible to use such representation for models such as additive trees. However, our previous techniques do not have such limitation since they are completely separated from the LTR. In this paper, we keep our discussion limited to linear models. We now define explore LTR (eLTR) function as follows:

$$y^e = f_e(\mathbf{X}) = g(f(\mathbf{X1}), \epsilon \times f(\mathbf{X2}))$$

where $y^e \in \mathbb{R}$ and $0 \leq \epsilon \leq 1$ is a variable in our algorithmic framework. Since, g is monotonic, $f_e(\mathbf{X}) <= f(X)$ when $\epsilon \leq 1$. Since feature set $X2$ is a biased feature set favoring old items, we can expect ranking based on f^e would be more in favor of new items in comparison with the original f, achieving the goal of emphasizing exploration of new items. Note that ϵ controls the amount of exploration: the smaller ϵ is, the more exploration (at the cost of exploitation). Since the maximum exploration is achieved when ϵ=0, in which case, ranking is entirely relying on f_1, the only loss in the original objective function is incurred by the removal of f_2. By controlling what features to be included in f_2, we can control the upper bound of the loss. In this sense, eLTR ensures a "safe" exploration strategy since f_1 is always active. Note, this function gradually can become very same as the LTR function when ϵ is close to 1. There can be various ways of constructing the ϵ such as the following:

eLTR basic exploration (eLTRb): In this strategy, we keep $\epsilon = \frac{I}{T_{max}}$. Here, I is an iteration and T_{max} is a maximum number of iteration after which everything can be reset. This is a very simple strategy where the eLTR just increases the importances of the behavioral features gradually with every iteration.

eLTR ucb weighted exploration (eLTRu): In this strategy, we keep $\epsilon = \frac{ucb_j}{U_j}$. Here, U_j is a normalization factor and in our experiment it is chosen to be the maximum UCB score in the set $E \cap R_i$.

eLTR ucb weighted exploration and reranking (eLTRur): This strategy first selects the top x items using eLTRu and it selects the remaining $(k-x)$ items using the classic LTR and then it reranks the k items using eLTRu.

Goswami et al. [12] show in their paper that eLTR can be considered as a monotonic submodular function and can be thus approximated by a greedy algorithm.

5.3 Adaptive submodular function based eLTR (AeLTR)

This is a extension of the eLTR mechanism presented by Goswami et al. [12]. However, it makes the strategy much more general by allowing us to use any LTR rank function going beyond the limitation of linear LTR models. In this,

we use the following function:

$$y^a = f_a(\mathbf{X}) = \epsilon_a f(\mathbf{X})$$

Here, the ϵ_a is a function of UCB as follows:

$$\epsilon_a = \frac{ucb_j}{U_j}$$

This algorithm has been discussed in a paper by Gabillon et al. [9]. We however, do not know how this strategy works for managing the e-commerce exploration.

6. EVALUATION METHODOLOGY

Due to the involvement of multiple objectives, the evaluation of E-Com search algorithms also presents new challenges. Here we discuss some ideas for evaluating the proposed e-LTR algorithm, which we hope will stimulate more work in this direction.

Given a set of queries and an initial ranking function, an e-LTR method is expected to "discover" the *ideal* ranking for each query, over a sequence of iterations. *Ideal* ranking may be defined as one that maximizes business metrics like sales, revenue etc. Since exploration is involved, one can expect the quality of ranking ($NDCG$) and consequently sales/revenue to fluctuate for individual queries during the discovery process. However, the benefit of an e-LTR method lies in its ability to deliver greater aggregate sales/revenue over some reasonable number of iterations N, compared to a non-discovery LTR method. We must thus compare our methods on the iterative growth in quality of ranking, and aggregate gain in business metrics.

The ideal approach for conducting such an evaluation would require simultaneously deploying all candidate methods to live user traffic, and computing various user engagement metrics such as click through rate,sales, revenue etc. However this strategy is difficult to implement in practice. Since user traffic received by each candidate method is different, we need to direct substantial amount of traffic to each method to make observations comparable and conclusions statistically significant. Deployment of a large number of experimental and likely sub-optimal ranking functions, especially when evaluating baselines, can result in significant business losses for e-Commerce search engines. More importantly, such an approach of using live user traffic would not allow us to have a fair comparison of any future algorithm with the ones that we test today, so it does not facilitate further study of the problem.

Perhaps a good and feasible strategy is to design a simulation-based counterfactual evaluation method [18]. Here, unlike a traditional static test collection, we also have to introduce models to simulate user behavior and model the biases observed in real system. This can be a topic itself and we omit further discussion on this as it is out of scope for this short paper.

7. EXPERIMENTAL RESULTS

In this section, we first construct a synthetic historical dataset with queries, items and their prices. We also generate the true purchase probabilities and utility scores for the item and query pairs. Additionally, we use a specific rank function to simulate the behavior of a trained LTR model.

Then we conduct a simulation as described in section **??** with various exploration strategies. During the simulation we use the observed purchase probabilities estimated from the purchase feedback as the most important feature for the rank function but we use the true probabilities generated during the initial data generation phase to simulate the user behavior.

The main goal of this experimental study is to evaluate the behavior of the exploration strategies with different distributions of the utility scores representing different state of the inventory in an e-com company.

We evaluate our algorithms by running the simulation for T times. We compute RPV and β-discoverability at the end of T iterations. We also compute a purchase based mean reciprocal ranking [6] metric (MRR). This metric is computed by summing the reciprocal ranks of all the items that are purchased in various user visits for all queries. Moreover, we also discretize our gold utility score between 1 to 5 and generate a rating for each item. This also allows us to compute a mean NDCG score at k-th position for all the search sessions as a relevance metric.

We expect to see that the RPV and NDCG of the LTR function will be the best. however the β-discoverability values will be better in ranking policies that use an exploration strategy. The new ranking strategies will incur loss in RPV and NDCG and based on our theoretical analysis we expect the eLTR methods to have less loss compared to the RSE and UCB based approaches in those measures. We also expect to see a loss in MRR for all exploration methods. However, we mainly interested in observing how these algorithms perform in β-discoverability metric compared to LTR.

7.1 Synthetic data generation

We first generate a set of N queries and M items. We then assign the prices of the items by randomly drawing a number between a minimum and a maximum price from a multi-modal Gaussian distribution that can have up to 1 to 10 peaks for a query. We select the specific number of peaks for a query uniform randomly. We also assign a subset of the peak prices to a query to be considered as the set of it's preferred prices. This makes a situation where every query may have a few preferred price peak points where it may also have the sales or revenue peaks.

Now that we have the items and queries defined, we randomly generate an utility score, denoted by (u_{ij}) for every item ζ_j for a query q_i. In our set up, we use uniform random, Gaussian and a long tailed distribution for selecting the utilities. These three different distributions represent three scenarios for a typical e-com company's inventory. Additionally, we generate a purchase probability between 0.02 to 0.40 for every item for every query. We generate these probabilities such that they correlates with the utility score. We generate these numbers in a way so that these are correlated with the utility scores with a statistically significant (p-value less than 0.10) Pearson correlation coefficient [24]. We also intend to correlate the purchase probability with the preferred peak prices for a query. Hence, we give an additive boost between 0 to 0.1 to the purchase probability in proportion to the absolute difference of the price of the item from the closest preferred mean price for that query. By generating the purchase probabilities in this way, we ensure that the actual purchase probabilities are related to the preferred prices for the queries, and also it is related to the utility scores of the items for a given query. Now, we de-

Algorithms	RPV	NDCG	MRR	$\beta - d$
LTR	0.12	0.96	0.54	0.10
RSE	0.06	0.76	0.21	0.78
UCBE	0.08	0.87	0.28	0.66
eLTRb	0.09	0.94	0.33	0.67
eLTRu	0.09	0.94	0.33	0.67
eLTRur	0.09	0.94	0.33	0.67
AeLTR	0.10	0.95	0.40	0.51

Table 1: Simulation of eLTR framework, with $|Q| = 100, |\mathcal{Z}| = 5000, |L| = 200, \beta - d = 10\%, \beta = 50, K = 6, x = 3, T = 50000$.

Algorithms	RPV	NDCG	MRR	$\beta - d$
LTR	9.56	0.57	0.45	0.11
RSE	7.55	0.27	0.25	0.76
UCBE	7.55	0.27	0.26	0.66
eLTRb	8.2	0.33	0.31	0.67
eLTRu	8.3	0.33	0.31	0.67
eLTRur	8.4	0.33	0.32	0.67
AeLTR	9.4	0.41	0.40	0.40

Table 2: Simulation of eLTR framework, with $|Q| = 100, |\mathcal{Z}| = 5000, |L| = 200, \beta - d = 10\%, \beta = 50, K = 6, x = 3, T = 50000$.

fine a β-discoverability rate $\beta = b$ and selects $b \times M$ items randomly from the set of all items. In our simulation, we assume that the estimated (observed) purchase probability for all the items in set E at the beginning can be zero. The rest of the items purchase probability are assumed to be estimated correctly at the beginning. Now, we create a simple rank function that is a weighted linear function of the utility score (u), the observed purchase probability (p_o), and the normalized absolute difference between the product price and the closest preferred mean price ($\hat{m}_p$ for the query such that $l = 0.60p_o + 0.20u + 0.20\hat{m}_p$. Here l denotes the score of the ranker. This ranker simulates a trained LTR function in e-com search where usually the sales can be considered the most valuable behavioral signal.

We now construct an user model. Here, an user browses through the search results one after another from the top and can purchase an item based on that item's purchase probability for that query. Note, in order to keep the simulation simple, we consider an user only purchases one item in one visit and leaves the site after that. We also can apply a discount to the probability of purchase logarithmically for each lower rank by multiplying $\frac{1}{\log_2(r+1)}$, where r is the ranking position of the item. This is to create an effect of the position bias [7].

7.2 Description of the experimental study

We conduct two sets of experiments with this simulated data.

In the first set of experiments we use a Gaussian distribution with mean 0.5 and the variance 0.1 for generating the utility scores, but everything else is same as the previous experiment. This means that the utility of the items in the inventory follows a normal distribution. We generate the scores for all the products from a Gaussian distribution with mean 0.5 and the variance 0.1. The table 1 shows the tables with final metrics for all the algorithms. We observe that with a Gaussian distribution of utility scores the eLTR approaches have better MRR, and β-discoverability. We observe that the AeLTR approach works well by producing good MRR and yet the β-discoverability is better than LTR. This is a powerful result since it shows that it is possible to integrate this with more sophisticated LTR algorithms

In the second set of experiment, we use a power law to generate the utility distribution. This means that only a small set of items here can be considered valuable in this scenario. The table 2 shows the final metrics for this case. We notice that even with this distribution of utility scores the eLTR variants have smaller loss in RPV, NDCG, and in MRR. Note that in this distribution, the discoverability can be considered to be naturally not so useful since a large number of items are not that valuable. We expect in such situation, a nice discoverability algorithm can help to eliminate items that do not get sold after sufficient exposure and enable the e-com company to optimize it's inventory. We observe a very similar trend for AeLTR algorithm as our previous experiment. It gives better RPV and MRR close to the LTR algorithm but it has less β-discoverability compared to eLTR approaches.

8. CONCLUSIONS AND FUTURE WORK

This paper represents a first step toward formalizing the emerging new E-Com search problem as an optimization problem with multiple objectives including the sales(CRV), revenue (RPV), and discoverability besides relevance. We formally define these objectives and discuss multiple strategies for conducting exploration with the learning to rank algorithms. We hope that our work will open up many new directions in research for optimizing e-com search. The obvious next step is to empirically validate the strategies based on log data from an e-com search engine. The theoretical framework also enables many interesting ways to further formalize the e-com search problem and develop new effective e-com search algorithms. Finally, the proposed discovery framework is just a small step toward solving the new problem of optimizing discoverability in e-com search; it is important to further develop more effective algorithms by using these algorithms as baselines.

9. REFERENCES

[1] S. Athey and D. Nekipelov. A structural model of sponsored search advertising auctions. In *Sixth ad auctions workshop*, volume 15, 2010.

[2] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[3] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM, 2005.

[4] C. J. Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.

[5] C. J. Burges, R. Ragno, and Q. V. Le. Learning to rank with nonsmooth cost functions. In *Advances in neural information processing systems*, pages 193–200, 2007.

[6] N. Craswell. Mean reciprocal rank. In *Encyclopedia of Database Systems*, pages 1703–1703. Springer, 2009.

[7] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, pages 87–94. ACM, 2008.

[8] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[9] V. Gabillon, B. Kveton, Z. Wen, B. Eriksson, and S. Muthukrishnan. Adaptive submodular maximization in bandit setting. In *Advances in Neural Information Processing Systems*, pages 2697–2705, 2013.

[10] A. Ghose and S. Yang. An empirical analysis of search engine advertising: Sponsored search in electronic markets. *Management Science*, 55(10):1605–1622, 2009.

[11] J. Gittins, K. Glazebrook, and R. Weber. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

[12] A. Goswami, C. Zhai, and P. Mohapatra. Learning to rank and discover for e-commerce search. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*.

[13] L. Hang. A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems*, 94(10):1854–1862, 2011.

[14] K. Järvelin and J. Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 41–48. ACM, 2000.

[15] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2002.

[16] S. K. Karmaker Santu, P. Sondhi, and C. Zhai. On application of learning to rank for e-commerce search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 475–484, New York, NY, USA, 2017. ACM.

[17] H. Li. A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems*, 94(10):1854–1862, 2011.

[18] L. Li, S. Chen, J. Kleban, and A. Gupta. Counterfactual estimation and optimization of click metrics in search engines: A case study. In *Proceedings of the 24th International Conference on World Wide Web*, pages 929–934. ACM, 2015.

[19] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[20] A. Mahajan and D. Teneketzis. Multi-armed bandit problems. In *Foundations and Applications of Sensor Management*, pages 121–151. Springer, 2008.

[21] D. Sorokina and E. Cantu-Paz. Amazon search: The joy of ranking products. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 459–460, New York, NY, USA, 2016. ACM.

[22] K. M. Svore, M. N. Volkovs, and C. J. Burges. Learning to rank with multiple objective functions. In *Proceedings of the 20th international conference on World wide web*, pages 367–376. ACM, 2011.

[23] J. Vermorel and M. Mohri. Multi-armed bandit algorithms and empirical evaluation. In *European conference on machine learning*, pages 437–448. Springer, 2005.

[24] R. R. Wilcox. *Introduction to robust estimation and hypothesis testing*. Academic press, 2011.

[25] Y. Yue, J. Broder, R. Kleinberg, and T. Joachims. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, 2012.