Learning to Rank and Discover for E-commerce Search

Anjan Goswami[1] and Chengxiang Zhai[2] and Prasant Mohapatra[1]

[1] University of California, Davis agoswami,pmohapatra@ucdavis.edu
[2] University of Illinois Urbana-Champaign cheng@uiuc.edu

Abstract. E-Commerce (E-Com) search is an emerging problem with multiple new challenges. One of the primary challenges constitutes optimizing it for relevance and revenue and simultaneously maintaining a discovery strategy. The problem requires designing novel strategies to systematically "discover" promising items from the inventory, that have not received sufficient exposure in search results while minimizing the loss of relevance and revenue because of that. To this end, we develop a formal framework for optimizing E-Com search and propose a novel epsilon-explore Learning to Rank (eLTR) paradigm that can be integrated with the traditional learning to rank (LTR) framework to explore new or less exposed items. The key idea is to decompose the ranking function into (1) a function of content-based features, (2) a function of behavioral features, and introduce a parameter epsilon to regulate their relative contributions. We further propose novel algorithms based on eLTR to improve the traditional LTR used in the current E-Com search engines by "forcing" exploration of a fixed number of items while limiting the relevance drop. We also show that eLTR can be considered to be monotonic sub-modular and thus we can design a greedy approximation algorithm with a theoretical guarantee. We conduct experiments with synthetic data and compare eLTR with a baseline random selection and an upper confidence bound (UCB) based exploration strategies. We show that eLTR is an efficient algorithm for such exploration. We expect that the formalization presented in this paper will lead to new research in the area of ranking problems for E-com marketplaces.

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 show the buyers a set of relevant and desirable products and facilitate the purchasing transactions that generate the revenue for the platform. Additionally, the e-com search also need to facilitate the discovery of the new or less exposed items to the buyers. This is in-fact critical for some categories such as apparel where new items are added periodically. However, a search ranking algorithm uses the behavioral signals such as sales, clicks, cart adds as features in its learning to rank algorithm. Therefore, the search engine may favor certain items that are purchased more by customers than other items in order to maximize the revenue,

but the more an engine favors certain items, the higher those items would be ranked in the search. This creates a conflict between the revenue and discovery metrics since some less-favored items might never have a chance of being exposed to the users. It is also easy to see maximizing discovery can compromise relevance of search results since those "unseen products" may not be relevant to a user's interest. We thus see that an e-com search engine must deal with a much more challenging optimization problem dealing with optimizing relevance and revenue as well as providing a discovery mechanism for the buyers. We address this problem in this paper and have made three contributions: Firstly, we suggest a formal framework for optimizing E-Com search and define multiple objectives to form a theoretical foundation for developing effective E-Com search algorithms. Secondly, we propose a simple and practical framework for conducting regulated discovery in e-com search. We then provide an exploration algorithm (eLTR) with a theoretical guarantee that can be easily integrated with traditional learning to rank algorithms. We also discuss how existing multi-armed bandit algorithms such as upper confidence bound (UCB) can also be used to address this problem in e-com search. Thirdly, we suggest a possible evaluation methodology based on simulation with E-Com search log data and show the effectiveness of our proposed eLTR algorithm using our evaluation methodology with synthetically generated data. We also compare different exploration strategies and show the effectiveness of eLTR algorithm.

2 Related work

There has been extensive research on learning to rank (LTR) algorithms particularly in the context of web search [11]. Most of the algorithms are designed to optimize a single metrics. Recently, Svore et al. [17] proposes a variant of LambdaMart [3] that can optimize multiple objectives particularly when two objectives are positively correlated. Authors conducted their experiments showing optimization of two different variants of normalized discounted cumulative gain (NDCG) [8] metrics that are based on judgments of human raters and based on click feedback respectively. In case of e-com search, aiming to maximize exploration can hurt the main business objectives and hence using this approach is not possible. The exploration algorithms are well researched in machine learning [16, 2, 7], particularly in the context of recommender systems [14, 15], news content optimization problems [10]. However, in e-com search we also need to ensure maximization of revenue and the exploration needs to be well regulated to minimize the expected loss. Vermorel et al. [18] in their paper compared the effectiveness of several multi-armed bandit (MAB) algorithms including heuristics such as ϵ-explore, soft-max etc., and also approaches based on upper confidence bound (UCB) [1] which has nice theoretical regret guarantee. The authors suggested often simple heuristics can provide very good practical performance for exploration. In this paper, we use a sub-modular function [12, 20] for exploration in order to have a nice theoretical guarantee for the exploration component. Our approach can be considered similar to the approaches used in learning adaptive

sub-modular function [6]. However, we integrated this with a ranking function and propose a novel function and prove the monotonic sub-modularity of it.

3 Optimization of e-com search

We consider the problem of optimizing an E-Com search engine over a fixed period of time $\{1, \cdots, T\}$. We assume that the search engine receives N queries, denoted by $Q = (q_1, q_2, \cdots, q_N)$ during this time. Let $\mathcal{Z} = \{\zeta_1, \cdots, \zeta_M\}$ be the set of M items during the same time. Let's denote the all the relevant items (recall set) for a query q_i by $R_i \subseteq \mathcal{Z}$. Consequently, we have $\mathcal{Z} = \bigcup_{i=1}^{N} R_i$. Now, we define a ranking policy by $\pi : 2^{(Q \times \mathcal{Z})} \rightarrow \Re$, where the input is the set of query item tuples where the items are from the recall set for the query and the output of the policy function is a subset of K items. These items are shown to the users and then an user browses the items one after another in the order they are shown. The user may click an item, add it to the shopping cart, and can also purchase. If a purchasing transaction happens then either the revenue generated or a sale can be considered to be a reward for the π. If the reward is designed to be using revenue it then needs to be real valued. If the reward is based on a sale, it can be a binary variable. It is also possible to construct the reward using clicks or cart-adds or a combination of all or some of these. An e-com search intends to maximize all these measures. However, the policy functions space is exponential and we require to formulate an optimization problem for e-com search. We don't generally have the knowledge when a purchasing transaction can happen. Hence, we introduce a binary random variable $\lambda_{ij} \in \{0, 1\}$ to indicate whether a purchasing transaction will happen with $\lambda_{ij} = 1$ meaning a purchasing event. Naturally, $p(\lambda_{ij} = 1|\zeta_j, q_i) + p(\lambda_{ij} = 0|\zeta_j, q_i) = 1$ for an item ζ_j shown for query q_i. The expected RPV for this query is then given by

$$RPV(q_t) = \sum_{\zeta_j \in \pi(q_i)} price(\zeta_j) \times N(\lambda_{ij} = 1|\zeta_j, q_i).$$

The total revenue defined on all the query results for the fixed period of time T when using policy π is thus

$$g_{RPV}(\pi) = \sum_{i=1}^{N} RPV(q_i)$$

Similarly, we can also define the relevance objective function as

$$g_{REL}(\pi) = \sum_{i=1}^{N} \rho(\pi(q_i))$$

where ρ can be any relevance measure such as nDCG, 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.

$$g_{REL}(\pi) = min_{i \in [1,p]} \rho(\pi(q_i))$$

We can now define the notion of discoverability of an e-commerce engine by considering a minimum number of impressions of items in a fixed period of time. The notion of discoverability is important because the use of 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. 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:

$$g_{\beta-discoverability} = \frac{\sum_{i=1}^{i=M} \gamma_i}{M}$$

Given these formal definitions, our overall optimization problem for the e-com search is to find an optimal ranking policy π that can simultaneously maximize all three objectives, i.e.,

$$\textbf{Maximize} \;\; g_{RPV}(\pi), g_{REL}(\pi), g_{\beta-dicoverability}$$

.

The above is a multi-objective problem and maximizing simultaneously all of the above objective may not be possible and it may also not be a desirable business goal from the platform side. The optimal tradeoff between the different objectives would inevitably application dependent.

The challenging aspect of this multi-objective problem is that the objectives such as discovery requires exploration that can also hurt the relevance and revenue.

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) methods 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 classic strategy is to use a convex combination of multiple objectives 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. One 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). Additionally, it does not have any exploration component and hence we can not ensure optimizing discovery with such algorithm.

4.2 Incremental Optimization

An alternative strategy is to take an existing LTR ranking function as a basis for a policy 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. We can then construct such a perturbation by keeping a fixed number of x positions for exploration out of the K top results. Then, the rest of the $(K - x)$ items can be selected using a LTR function based on other criteria such as combination of revenue and relevance. This framework is so simple that it is very easy to realize in practice but it is possible to conduct exploration based on several strategies such that the regret in the form of loss of revenue and relevance can be minimized. In the next section of the paper, we discuss a few such strategies.

5 Exploration with LTR (eLTR)

Let us 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.

Now, we propose 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

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. In this paper, we experimented with three different expressions for ϵ. These are as follows:

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 $E \cap R_i$. This can be intuitively considered as the expected LTR score based on a sales estimation. It is motivated by adaptive sub-modular optimization in bandit setting [6] that has nice regret guarantee.

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.

6 Theoretical analysis

In this section, we discuss the regret bounds of all the strategies. We express the regret in terms of total number of search session n in a fixed period of time T. Our first strategy RSE can be arbitrarily bad and can have a worst case regret proportional to $O(xn)$. However, it can have a fast discovery. The UCB is a better strategy compared to RSE. The regret of stochastic variant of UCB can be estimated as $O(\log(xn))$. The discovery in this algorithm will be not as good as the RSE and it can be worst if the MAB arms converge fast towards optimality. On the other hand, we can construct the eLTR function as monotonic sub-modular. Then, the regret for eLTR can be estimated as $O(1+1/e^{-\frac{|E|}{x}})$ times worst compared to the optimal. In our case, the optimal algorithm is LTR [13]. The eLTR algorithm is inspired from ϵ-greedy style MAB algorithm and hence can have better discovery compared to UCB. However, it is not clear if the regret is necessarily better than UCB based strategy. In section 8 we conduct a simulation to understand how these algorithms compare with each other. Here, we now show that eLTR can be indeed monotonic sub-modular.

6.1 Monotonic sub-modularity of eLTR

Let's call the ranking policy for selecting x items from the set E as

$$\pi^e : 2^{(Q \times \mathcal{E})} \rightarrow \Re$$

, where the cost function for our policy can be as follows:

$$c(E) = arg \max_{\zeta \in E} \sum_{i=1}^{i=x} y_i^e$$

We now show that this cost function is monotonic.

If we add a new item in set E, that will be added to the result of a query if the eLTR score for that query and that item is greater than the score of existing top x items. In that case, the cost of eLTR will increase. If the eLTR score for the query and the item is less than the existing top x items, the overall score from eLTR will be unchanged. Hence, the function is monotonically nondecreasing.

Now, we show that this function is sub-modular.

Let us assume that $A \subset B \subseteq E$. Let's also assume that there is an item $\zeta_g \notin (A \cup B)$. Consequently, we can have, $a = c(A \cup \{\zeta_g\}) - c(A)$ and $b = c(B \cup \{\zeta_g\}) - c(B)$.
There can then be three cases: case 1: $c(B) \geq c(A)$
In this case, there must be one or more high eLTR items in set B. Now, if we add the item ζ_g, it will either get added to the top x or not. If it is added to the top items in set B, that means it replaces at least the one item with the minimum eLTR score in top x items in set B. If there are no common items in the top x items for A and B, and since $c(B) \geq c(A)$, the new item has a higher eLTR value than any items in set A and will also replace an item in top x for A. Hence, $a = b$.
Now, if the item does not get added to top x items in B, that means the item does not have higher value compared to the top x items in B. Then, we have $b = 0$. Now, the item can be added in top items for A or not. If it is added in A then we will have $a > 0$ and if it is not added then we have $a = 0$.

case 2: $c(B) \leq c(A)$
This case will never happen since all the items in set A are also in set B and if there are top items in set A, all of those items will be in set B. Hence, unless there are items with higher eLTR compared to the top items in A, top items in B will never be different.

case 3: $c(B) = c(A)$
This is the simplest case. All top items are same and the new item will either get added in both or not since it need to replace one of the top items. Hence, $a = b$.

We have now shown that this cost function always have $a >= b$ and hence this function is sub-modular.

7 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. 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. Perhaps a good and feasible strategy is to design a simulation-based evaluation method using counterfactual techniques [9]. Here, we use historical search session data to replay the sessions for a fixed period of time. We then artificially make a set of items selected randomly as candidates for exploration where we do not have estimation of purchase probabilities. We keep these items in set E. We use the true purchase probabilities estimated from the data fro the items that have been shown sufficient number of time in our rank function but use zero values for the same probabilities for the items in set E.

On the surface, it appears that we may simply use the clicks or sales of the items to estimate the utility of each product. However, such a commonly used strategy would inherently favor already exposed items, and if an item has never been exposed, its utility would be zero, thus this strategy cannot be used for evaluating discoverability. To ensure discoverability for potentially *every* item in the collection, we can define the gold utility of a query product pair $u_{q,d}$ as a number randomly sampled between $[0, 1]$. Such a random sampling strategy would give every item a chance of being the underexposed target to be "discovered." Thus although the assigned utilities in this way may not reflect accurately real user preferences, the simulated utility can actually give more meaningful evaluation results than using click-throughs to simulate utility when comparing different exploration-exploitation methods where only the relative difference of these methods matters.

8 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 7 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 (a) with various different sets of number of queries and number of items, (b) with different values of β-discoverability at the beginning, (c) 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 [4] 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.

8.1 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 [19]. 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 define 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 [5].

8.2 Description of the experimental study

We conduct four sets of experiments with this simulated data.

In the first set of experiments, we use a small set of queries and a small set of products to understand the nature of the algorithms. The utility scores for all the products are generated from an uniform random distribution.

The table 1 shows the RPV, NDCG@6,PMRR, and β-discoverability. We note that all the variants achieve high discoverability score with relatively small loss in RPV, NDCG and MRR. It is clear that eLTRur performs better than all other approaches. It in-fact performs even better than the LTR algorithm in RPV metric along with doing well in discovery.

In the second set of experiments, we use a larger number of queries and products and select a smaller starting value for β-discoverability. We also run this simulation longer. In table 2, we find the eLTR variants perform much better compared to the UCBE, and RSE. In-fact, this time eLTR variants also perform as good as the LTR also in NDCG metric.

In the third 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. We again see in table 3 that eLTR variants perform well compared to UCBE and RSE and they also do better in terms of

Algorithms	RPV	NDCG	MRR	$\beta - d$
LTR	0.09	0.94	0.41	0.37
RSE	0.089	0.86	0.38	0.97
UCBE	0.09	0.87	0.39	0.96
eLTRb	0.09	0.88	0.39	0.97
eLTRu	0.09	0.88	0.39	0.97
eLTRur	0.092	0.88	0.40	0.98

Table 1: Simulation of eLTR framework, with $|Q| = 10, |\mathcal{Z}| = 100, |L| = 50, \beta - d = 20\%, \beta = 50, K = 6, x = 3, T = 10000$.

Algorithms	RPV	NDCG	MRR	$\beta - d$
LTR	0.12	0.90	0.42	0.12
RSE	0.09	0.73	0.27	0.30
UCBE	0.10	0.73	0.27 0.66	
eLTRb	0.11	0.91	0.32	0.68
eLTRu	0.11	0.92	0.32	0.68
eLTRur	0.11	0.92	0.32	0.68

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

NDCG compared to LTR. The table **??** shows the convergence plots for the six competing algorithms for RPV, MRR, and the discovery.

kept all the parameters same except the distribution of utility score. We generate the scores for all the products from a Gaussian distribution with mean 0.5 and the variance 0.1. The table 3 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.

Algorithms	RPV	NDCG	MRR	$\beta - d$
LTR	0.10	0.92	0.44	0.10
RSE	0.08	0.86	0.28	0.29
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

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

In the fourth 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 5 shows the final metrics for this case and

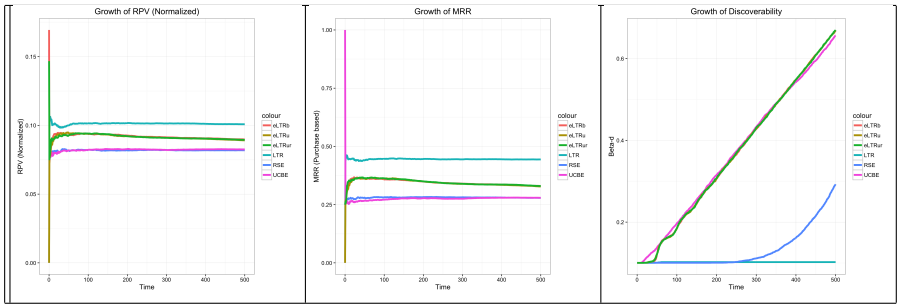

Table 4: Simulation of eLTR framework, with $|Q| = 100, |\mathcal{Z}| = 5000, |L| = 200, \beta - d = 10\%, \beta = 50, K = 6, x = 3, T = 50000$.

the figure 6 shows the convergence plots for RPV, NDCG and discoverability for the six different algorithms. 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. The table 6 shows the convergence plots of all the algorithms in this scenario.

Algorithms	RPV	NDCG	MRR	$\beta - d$
LTR	9.56	0.57	0.45	0.11
RSE	7.55	0.27	0.25	0.30
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

Table 5: Simulation of eLTR framework, with $|Q| = 100, |\mathcal{Z}| = 5000, |L| = 200, \beta - d = 10\%, \beta = 50, K = 6, x = 3, T = 50000$.

9 Conclusions

This paper represents a first step toward formalizing the emerging new E-Com search problem as an optimization problem with multiple objectives including the revenue per-visit (RPV), and discoverability besides relevance. We formally define these objectives and discuss multiple strategies for solving such an optimization problem by extending existing learning to rank algorithms. We also proposed a novel exploratory Learning to Rank (eLTR) method that can be

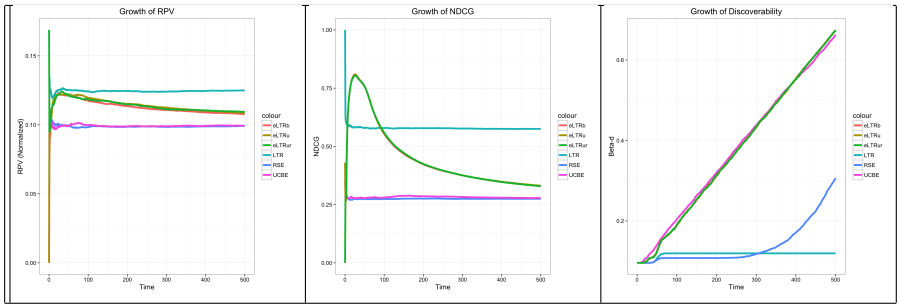

Table 6: Simulation of eLTR framework, with $|Q| = 100, |\mathcal{Z}| = 5000, |L| = 200, \beta - d = 10\%, \beta = 50, K = 6, x = 3, T = 50000$ and with Pareto distribution for the utility scores and the purchase probabilities.

integrated with the traditional LTR framework to explore new or less exposed items and discussed possible methods for evaluating eLTR. We show that selecting the items from a set of yet not discovered items using eLTR can be mapped to a monotonic sub-modular function and hence the greedy algorithm has nice approximation guarantees.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 proposed eLTR strategy by using the proposed simulation strategy based on log data from an e-com search engine. The proposed theoretical framework also enables many interesting ways to further formalize the e-com search problem and develop new effective e-com search algorithms based on existing multi-armed bandit and sub-modular optimization theories. Finally, the proposed eLTR algorithm 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 that can be applied with non-linear learning to rank algorithms.

References

1. Auer, P., Ortner, R.: Ucb revisited: Improved regret bounds for the stochastic multi-armed bandit problem. Periodica Mathematica Hungarica 61(1-2), 55–65 (2010)
2. Bubeck, S., Cesa-Bianchi, N.: Regret analysis of stochastic and nonstochastic multi-armed bandit problems. FOUNDATIONS AND TRENDS IN MACHINE LEARNING 5(1), 1–122 (2012)
3. Burges, C.J.: From ranknet to lambdarank to lambdamart: An overview. Learning 11(23-581), 81 (2010)
4. Craswell, N.: Mean reciprocal rank. In: Encyclopedia of Database Systems, pp. 1703–1703. Springer (2009)
5. Craswell, N., Zoeter, O., Taylor, M., Ramsey, B.: An experimental comparison of click position-bias models. In: Proceedings of the 2008 International Conference on Web Search and Data Mining. pp. 87–94. WSDM '08, ACM (2008)

6. Gabillon, V., Kveton, B., Wen, Z., Eriksson, B., Muthukrishnan, S.: Adaptive submodular maximization in bandit setting. In: Advances in Neural Information Processing Systems. pp. 2697–2705 (2013)
7. Gittins, J., Glazebrook, K., Weber, R.: Multi-armed bandit allocation indices. John Wiley & Sons (2011)
8. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of ir techniques. ACM Transactions on Information Systems (TOIS) 20(4), 422–446 (2002)
9. Li, L., Chen, S., Kleban, J., Gupta, A.: Counterfactual estimation and optimization of click metrics in search engines: A case study. In: Proceedings of the 24th International Conference on World Wide Web. pp. 929–934. ACM (2015)
10. Li, L., Chu, W., Langford, J., Schapire, R.E.: A contextual-bandit approach to personalized news article recommendation. In: Proceedings of the 19th international conference on World wide web. pp. 661–670. ACM (2010)
11. Liu, T.Y.: Learning to rank for information retrieval. Foundations and Trends in Information Retrieval 3(3), 225–331 (2009)
12. Lovász, L.: Submodular functions and convexity. In: Mathematical Programming The State of the Art, pp. 235–257. Springer (1983)
13. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functionsi. Mathematical Programming 14(1), 265–294 (1978)
14. Park, S.T., Chu, W.: Pairwise preference regression for cold-start recommendation. In: Proceedings of the third ACM conference on Recommender systems. pp. 21–28. ACM (2009)
15. Schein, A.I., Popescul, A., Ungar, L.H., Pennock, D.M.: Methods and metrics for cold-start recommendations. In: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 253–260. ACM (2002)
16. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction, vol. 1. MIT press Cambridge (1998)
17. Svore, K.M., Volkovs, M.N., Burges, C.J.: Learning to rank with multiple objective functions. In: Proceedings of the 20th international conference on World wide web. pp. 367–376. ACM (2011)
18. Vermorel, J., Mohri, M.: Multi-armed bandit algorithms and empirical evaluation. In: ECML. vol. 3720, pp. 437–448. Springer (2005)
19. Wilcox, R.R.: Introduction to robust estimation and hypothesis testing. Academic press (2011)
20. Yue, Y., Guestrin, C.: Linear submodular bandits and their application to diversified retrieval. In: Advances in Neural Information Processing Systems. pp. 2483–2491 (2011)