Gathering Additional Feedback on Search Results by Multi-Armed Bandits with Respect to Production Ranking

Aleksandr Vorobev[1], Damien Lefortier[1,2], Gleb Gusev[1], and Pavel Serdyukov[1]

[1]Yandex, Moscow, Russia,
{alvor88,damien,gleb57,pavser}@yandex-team.ru
[2]University of Amsterdam, Amsterdam, The Netherlands

ABSTRACT

Given a repeatedly issued query and a document with a not-yet-confirmed potential to satisfy the users' needs, a search system should place this document on a high position in order to gather user feedback and obtain a more confident estimate of the document utility. On the other hand, the main objective of the search system is to maximize expected user satisfaction over a rather long period, what requires showing more relevant documents on average. The state-of-the-art approaches to solving this exploration-exploitation dilemma rely on strongly simplified settings making these approaches infeasible in practice. We improve the most flexible and pragmatic of them to handle some actual practical issues. The first one is utilizing prior information about queries and documents, the second is combining bandit-based learning approaches with a default production ranking algorithm. We show experimentally that our framework enables to significantly improve the ranking of a leading commercial search engine.

1. INTRODUCTION

A common search system is usually based on a deterministic ranking model that aggregates both *pre-feedback* features describing the content of web pages and *implicit feedback* features based on user behavior data stored in query logs. This leads to the following iterative process of interaction with users that repeatedly submit a particular query. At the first stage, when the query is relatively new to the system, it ranks documents by the scores using their pre-feedback information only. Further, at the second stage, it corrects this ranking with respect to implicit feedback data, while it is being collected. During this stabilizing phase, scores of top ranked documents which get negative user feedback become lower, so these documents are exchanged with other documents with high pre-feedback based scores. After the algorithm found enough documents getting mostly positive user feedback, the ranking is not being changed anymore by two reasons: first, the algorithm continues to receive only redundant confirmation of the top documents' relatively high relevance, and, second, no documents lacking implicit feedback have scores higher than those which were lucky to get some. At the same time, pre-feedback information cannot fully reflect all the aspects of the documents that potentially impact user satisfaction, therefore, some of those documents lacking user feedback can be more relevant than those ranked higher. However, the customary scheme of user–system interaction just described cannot find the additional evidence for that, since these low ranked documents are very unlikely to ever receive user feedback. Therefore, it may be of use to have other mechanisms to place some lower ranked documents on the upper positions to attract more user feedback to them. In this way, we may degrade the query performance for a short period of time, by taking the risk of showing some less relevant documents on the top positions, but improve it in the long term, by giving a chance to get user feedback (and, hence, improve their scores) for more potentially highly relevant documents.

Thus, there are two different ranking strategies: the exploitative one, aiming, at each step, to maximize the ranking performance for the current query issue , and the exploratory one, allowing to collect more user feedback on lower ranked documents even at the cost of degrading the ranking performance for some query issues. It is especially important to achieve the optimal interplay between these two strategies that maximizes the cumulative quality of a series of consecutive query issues. We refer to this problem as *Online Learning to Rank with Exploration and Exploitation (OLREE)* problem in this paper. It is a particular case of the well-known exploration–exploitation trade-off problem formalized most appropriately in the Stochastic Multi-Armed Bandit (SMAB) setting [2]. This setting originates from a problem of a gambler facing a row of casino slot machines, sometimes known as "one-armed bandits". Here we briefly describe the problem setup.

There is a set of *arms* $A = \{a_i\}$. At each *step* t, an algorithm chooses an arm $a(t) \in A$, receives its *reward* R_t, which is a sample from an unknown arm-associated distribution with an unknown expectation $r_{a(t)}$ (we call r_a the arm *gain* throughout the paper), and updates the stored information about $a(t)$. The algorithm goal is to maximize the cumulative reward $\mathbf{R}(T)$ over the first T steps: $\mathbf{R}(T) = \mathbf{E}\left(\sum_{t=1}^{T} R_t\right)$, where $\mathbf{E}$ denotes expectation. The general idea underlying

WWW 2015, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3469-3/15/05.
http://dx.doi.org/10.1145/2736277.2741104.

a bandit algorithm is the following. It stores information about each arm a, which includes not only an estimate of r_a on the basis of the observed user behaviour, but also the confidence in this estimate which can be represented, for example, as its standard deviation or as a distribution over values of r_a. Then, at each step, an algorithm balances between a purely exploratory choice (the arm with the lowest confidence) and a purely exploitative choice (the arm with the highest estimated r_a).

In this paper, we investigate one of the most promising ways to formulate the OLREE problem in the SMAB terms. We consider a repeatedly submitted query as a separate bandit problem setting, each query submission as one step, a ranking system as a gambler, and each document as an arm. Examination of a document by a user is considered as pulling of the corresponding arm, and the reward of the arm is then determined by the user satisfaction with the examined document. As we discuss in Section 8, this approach allows us to apply SMAB theory more appropriately in practice than do the other known applications of this theory to OLREE. Unlike in the original SMAB formulation, within this approach, a gambler is able to choose not one but several arms for pulling. Their number is upper-bounded by the capacity of the search engine result page (SERP). However, after an arm is chosen for pulling (so, a document is included into the ranking), the gambler has no control over it and whether it will be eventually pulled or not depends entirely on the user who issued the query. Moreover, due to the limitations of logging mechanisms, while estimating the gain r_d of a document d and the confidence in this estimate, the gambler does not often know whether the pulling of the arm actually took place. In this case, the SMAB formulation and the known bandit algorithms cannot be directly applied to the OLREE task and should be adopted to it.

To the best of our knowledge, the only existing adoption that attempts to solve the mentioned problems was suggested by Sloan and Wang [31]. It is based on utilizing a click model to infer information about the document gain from the user behavior and takes position bias into account. However, this pioneering study did not address another problem, whose solution would actually make this promising "document=arm" approach practically useful: no real-world search engine can afford to ignore the initial guess about the document relevance made on the ground of the *prior* information that is already available before the exploration starts. Neither it is reasonable and feasible to explore all documents available to the search system, as such intense exploration will severely degrade user experience, what is highly undesirable even for a very short period. Thus, we need to explore only the documents, in whose scores the ranking system is least confident. The prior estimation of both r_d and our confidence in this estimate and smooth introduction of a bandit-based ranking algorithm into a default ranking system are the essential problems that have not been addressed so far.

In this paper, we propose an approach to marry a state-of-the-art default ranking model with the above-described bandit-based formulation. Particularly, we develop an approach for setting prior values of r_d estimates and of confidences in them based on both the document features available to the default ranking system and the ranked list of documents it produces. This approach relies on two techniques that have not been previously applied to the OLREE problem: constructing a regression model for error prediction and applying isotonic regression to adjust regression to ranking. We show experimentally that incorporation of our bandit-based algorithm into a highly optimized production ranker of a major commercial search engine remarkably increases the performance of the latter during a 10-day period.

To sum up, our contributions are:

- A new formulation of the OLREE problem in the terms of the SMAB problem (Section 2).
- An approach to utilizing prior information in the OLREE problem by estimating prior distributions of documents' relevances (i.e., the confidences in their prior estimates) (Section 4),
- An algorithm for combining a state-of-the-art ranking model with a bandit-based ranking approach (Section 5).

Besides, we present experimental results in Section 7, review related work on application of bandit-based methods to IR tasks in Section 8, and make conclusions in Section 9.

2. PROBLEM FORMALIZATION

We suggest the following SMAB formulation for the OLREE problem which leads to the optimization problem considered in [31, Eq. 1]. Given a unique query, we associate it with a dedicated bandit algorithm which makes a step per each query issue. This algorithm treats each document related to the query as an arm. At each step, in order to choose the list of arms for pulling, the algorithm generates a SERP in response to the current query issue and observes some user behavior on it. The main difference of our OLREE setting from the ordinary SMAB problem described in Section 1 and from the approaches of all the related studies, except for [31], is that an arm chosen by the algorithm for pulling can be both *pulled* or *not pulled* depending on whether the user, who issued the query, actually *examined* or *did not examine* the corresponding document on the SERP (similar events $\{S_i = j\}, j = 1, 2$ were used in [31, Sec. 3.2]). This leads to the notion of document *trial* which means the fact of pulling the corresponding arm. Each examined document provides a reward to the algorithm, which equals 1, if the user is *satisfied* with the document, and 0 otherwise (*unsatisfied*). We also assume that the reward of a document d is a Bernoulli random variable with a probability of success r_d.

In the scope of this paper, we define satisfaction as a click with a long enough dwell time or the last click on the SERP (see Section 6.2 for details). In this way, the cumulative reward of our bandit algorithm over the first T query issues equals the cumulative number of satisfied clicks, as in [31] (in their study, each click was considered to be satisfied). We also assume that user behavior on search result pages is consistent with the following conditions (which are implied by the examination hypothesis [11]): a search result can be clicked by the user only if it is examined by her, the examination probability does not depend on the document, and, given the fact of examination, the click probability depends on the document only. So, it is reasonable to consider that an arm is pulled if and only if the user examined the snippet of the corresponding document. Thus, the cumulative reward over T issues of a query could be represented as

$$\mathbf{R}(T) = \sum_{t=1}^{T} \sum_{i=1}^{k} P(E^t_{d_i(t)} = 1) P(C^t_{d_i(t)} = 1 | E^t_{d_i(t)} = 1),$$

where the binary variables E^t_d and C^t_d indicate the examination of the document snippet ($E^t_d = 1$) and the satisfied

click on the document ($C_d^t = 1$) at step t respectively, k is the number of documents on each SERP, and $d_i(t)$ is the document placed on position i by our algorithm at step t.

As long as one can never be sure that the user will examine a particular document, unless it is shown on the first position, the number of document trials that will actually take place during the period of T query issues is a random variable. It is the principal difference of our approach and that of [31] (their "effective" number of impressions is similar to our number of trials, see their Section 3.2) from the other SMAB formulations of the OLREE problem we are familiar with.

On the other hand, SMAB algorithms essentially use the information of which arms were pulled. Therefore, their application to the OLREE problem requires to infer which snippets on the SERP were examined. In Section 3.3, we describe both a binary inference approach, which unambiguously defines if the snippet was examined or not, and a probabilistic inference approach, which obtains the probability of examination and performs better in practice.

3. APPROACH

3.1 Background on SMAB algorithms

Our approach to OLREE problem relies on a SMAB algorithm B of the following type. In the course of the game, it stores a vector of real-valued parameters $F_t(a) \in \mathbb{R}^I$ for each arm a, where t denotes the step number, $\mathbb{R}^I$ is an algorithm-specific parameter space and I is the set of parameter indices. At each step t, algorithm B calculates scores

$$S_t(a) := S_{B,t}(F_t(a)),$$

where $S_{B,t}$ is either a random or deterministic scoring function on $\mathbb{R}^p$ depending on particular algorithm B. Algorithm B chooses an arm $a_t^* \in \arg\max_a(S_t(a))$, observes its reward, and calculates $F_{t+1}(a_t^*)$ according to some rule U_B on the basis of $F_t(a_t^*)$ and the observed reward. It sets $F_{t+1}(a) = F_t(a)$ for other arms a.

In the next two sections, we describe two SMAB algorithms we adopt to the OLREE problem in Section 3.2. Below we also describe a class of SMAB algorithms, which can be easily applied within our approach.

3.1.1 UCB-1

First, we adopt the state-of-the-art algorithm **UCB-1** [2], which was applied to the OLREE problem in the main related studies [27, 31], where it outperformed its competitors. Following UCB-1, at step t,

$$S_t(a) = \widehat{r}_{a,t} + \alpha\sqrt{\frac{2ln\ t}{\gamma_{a,t}}}, \quad (1)$$

where $\widehat{r}_{a,t} = \frac{W_{a,t}}{\gamma_{a,t}}$ is the maximum likelihood estimate of r_a after step $(t-1)$, $\gamma_{a,t}$ is the number of trials of the arm a during the first $(t-1)$ steps, $W_{a,t}$ is the number of successful trials among them, α is a parameter controlling the exploration rate (to be fitted on a training data). Thus, $F_t(a)$ consists of two components, $W_{a,t}$ and $\gamma_{a,t}$. The rule U_B increments $\gamma_{a,t}$ by 1 when a is pulled ($\gamma_{a,t+1} = \gamma_{a,t} + 1$) and increments $W_{a,t}$ by 1 when this pulling is successful.

3.1.2 Bayesian bandits

In the case of the standard SMAB problem, bandit algorithms following the Bayesian approach, namely, Bayesian-UCB [19] and Thompson sampling [20], have stronger theoretically grounded guarantees and achieved better performance in experiments with other tasks [19, 20] than the pointwise alternatives such as UCB-1. However, to the best of our knowledge, they have not been applied to the OLREE problem earlier. We further describe *Bayesian bandits* algorithm, which generalizes both Bayesian-UCB and Thompson sampling. It relies on the posterior probability density function $p_{a,t}(r), r \in [0,1]$, of gain r_a. This posterior probability is updated by rule U_B to be specified below. At step t, algorithm samples α uniformly from the set $[\alpha_{low}(t), \alpha_{up}(t)]$ ($\alpha_{low}(t)$ and $\alpha_{up}(t)$ are parameters of the algorithm) and then sets $S_t(a)$ to the α-quantile of $p_{a,t}(r)$ (i.e., $\int_0^{S_t(a)} p_{a,t}(r)dr = \alpha$). Thus, $F_t(a)$ coincides with $\{p_{a,t}(r)\}_{r\in[0,1]}$. At step t, with observed reward R_t, rule U_B keeps $p_{a,t+1}(r) = p_{a,t}(r)$ for $a \neq a_t^*$ and updates $p_{a,t}(r)$ for $a = a_t^*$ as follows:

$$\begin{aligned} p_{a,t+1}(r) = p_{a,t}(r|R_t) \propto P(R_t|r_a = r)p_{a,t}(r) = \\ = r^{R_t}(1-r)^{1-R_t}p_{a,t}(r) \propto \\ \propto r^{W_{a,t+1}}(1-r)^{\gamma_{a,t+1}-W_{a,t+1}}p_{a,0}(r). \quad (2) \end{aligned}$$

Thus, Bayesian bandit, in contrast to UCB-1, requires some initialization, i.e., prior distribution $p_{a,0}(r)$ for each arm a. In the case of $\alpha_{low}(t) = \alpha_{up}(t)$, this algorithm coincides with Bayesian-UCB [19], and if $\alpha_{low}(t) = 0, \alpha_{up}(t) = 1$, it reduces to Thompson sampling [20]. Note that we can store $\{\gamma_{a,t}, W_{a,t}, \{p_{a,0}(r)\}_{r\in[0,1]}\}$ instead of $\{p_{a,t}(r)\}_{r\in[0,1]}$ and update only $\gamma_{a,t}$ and $W_{a,t}$ when we run Bayesian bandits in practice.

Since the SMAB problem is to maximize the expectation of the cumulative reward, the rational exploitative strategy is to choose the arm a with the maximum mean of the posterior distribution $p_{a,t}(r)$. This motivated us to consider the following modifications of Bayesian bandits and UCB-1 (we call them MeanBayes and MeanUCB-1 respectively) whose exploitative settings rank documents by the posterior mean values. The algorithm MeanBayes uses the following scoring function similar to Equation 1: $S_t(a) = \overline{r}_{a,t} + \alpha \cdot \sigma_{a,t}$, where $\overline{r}_{a,t}$ and $\sigma_{a,t}$ are the mean and the standard deviation of the posterior distribution $p_{a,t}(r)$ respectively.

The modification MeanUCB-1 calculates $\widehat{r}_{a,t}$ by the following modified rule: $\widehat{r}_{a,t} = \frac{W_{a,t}+1}{\gamma_{a,t}+2}$. While the original rule takes the maximum likelihood estimate of r_a, which corresponds to the mode of the posterior distribution $p_{a,t}(r)$, the estimate of the modified rule corresponds to its mean in the assumption that $p_{a,t}(r)$ is a beta distribution (see Section 4).

Now we describe our general approach to the OLREE problem, which is capable of adopting a variety of known SMAB algorithms, including the above-mentioned methods.

3.2 General Approach to OLREE

In our framework, we assume that any algorithm solving the OLREE problem (*OLREE algorithm* below) works for each query independently and deals with a limited set of candidate documents which are chosen for exploration on the ground of the pre-feedback information.

Adopting any SMAB algorithm B to the OLREE problem requires answers to two questions: how to choose not one but the list of documents for the SERP and how to update parameters $F_t(d)$ of each document on the ground of the user feedback. The first problem has a very natural solution

which is to rank documents by their scores $S_t(d)$ at step t. In order to solve the second problem, Sloan and Wang [31] suggested to update $F_t(d)$ by a specific rule U which, for each document d from the SERP, estimates the posterior probability that d was examined, relying on a click model. Then, this probability is considered as a weight of the document trial taken into account when updating $F_t(d)$. It is obviously implied that $F_{t+1}(d) = F_t(d)$ for each document d not presented on the SERP at step t. We call this adoption of the SMAB algorithm B (see Algorithm 1) as a **Bandit-Based Ranking Algorithm (BBRA)**.

Algorithm 1: Bandit-Based Ranking Algorithm (BBRA)

Data: set of documents D for a given query, number of query issues T, prior $F_0(d)$ for each $d \in D$;
for $t = 0$ **to** $T - 1$ **do**
  foreach $d \in D$ **do**
    $S_t(d) = S_{B,t}(F_t(d))$;
  end
  Sort D by decreasing S_t: $\{d_1(t), d_2(t), \ldots\}$;
  Display documents $\{d_1(t), \ldots, d_k(t)\}$ to the user;
  Observe user behavior C on the SERP;
  for $i = 1$ **to** k **do**
    $F_{t+1}(d_i(t)) = U(F_t(d_i(t)), C)$;
  end
end
Result: Ranking for each query issue: $\{d_i(t)\}_{i=1..k}^{t=1..T}$

Now we describe the specific rules U for updating parameters $F_t(d)$ utilized by each of the SMAB scoring functions studied in this paper. Note that any SMAB algorithm with parameters $F_t(d)$ which could be calculated on the basis of $\left\{W_{a,t}, \gamma_{a,t}, \{p_{a,t}(r)\}_{r\in[0,1]}\right\}$ can easily be used within BBRA with the below-described update rules.

3.3 Update rule for OLREE

All the updated rules we experiment with rely on the Dependent Click Model (DCM) [15], since it was used in different studies on bandit algorithms in web search ranking [17, 31] and outperformed the alternative models in the experiments in [31]. To replace DCM with any other click model, one should specify an update rule based on it. However, any of the below-described general methods for constructing update rules (EM-algorithm for updating parameters of UCB-1, Bayesian inference for Bayesian bandits) can be applied to any click model.

As most of state-of-the-art click models, DCM is based on the cascade hypothesis. In terms of variables $d_i(t), E_d^t$ and C_d^t introduced in Section 2, DCM is described by the following equations:

$$P(E^t_{d_1(t)} = 1) = 1,\ P(E^t_{d_{i+1}(t)} = 1|\ C^t_{d_i(t)} = 1) = \lambda_i,$$
$$P(E^t_{d_{i+1}(t)} = 1|\ E^t_{d_i(t)} = 1, C^t_{d_i(t)} = 0) = 1,$$
$$P(C^t_d = 1|E^t_d = 1) = r_d. \quad (3)$$

Naturally, we assume that satisfied clicks (i.e., C_d^t) are directly observed by the search engine, and examinations (i.e., E_d^t) are not, as it occurs in practice. Thus, E_d^t are hidden variables of the constructed Bayesian network. As in [15], we do not infer anything from abandoned sessions with no clicks, because such sessions cannot be explained by DCM in the case of real user behavior and the reasons of abandonment are often unknown. The cascade hypothesis allows to regard the documents above (and on) the lowest click position as examined, while updating parameters $F_t(d)$ on the ground of the user feedback. Further, we use two approaches to estimate the values E_d^t and, on the basis of these estimates, to update parameters $F_t(d)$ for the documents d placed below the lowest click. Within the first, *honest*, approach, for each of such documents, we follow [31] and estimate the posterior probability that it was examined. When updating $F_t(d)$, we use this probability as the weight of the document trial (see Equation 4). The second one, *negligent*, is to regard these documents as not examined by the user. It takes into account less information but requires much less computations than does the honest approach. We experiment with both approaches to verify if the complex computations of the latter one are indeed justified by its performance. Specifically, we have the following update rules U.

1) The **negligent** approach unambiguously defines which documents were examined at each step t. Thus, we can just use rule U_B (corresponding to the chosen SMAB algorithm B described in Section 3.1) to update $F_t(d)$ for each examined d. In terms of that section, $\gamma_{d,t} = |\{\tau < t : E_d^\tau = 1\}|$, $W_{d,t} = |\{\tau < t : C_d^\tau = 1\}|$ (here $|A|$ is the size of set A).

2) Now we describe the update rules for UCB-1 and Bayesian bandits specifically (which are also applicable to their modifications MeanUCB-1 and MeanBayes) in the case of the **honest** inference approach.

- The **point estimate** $\widehat{r}_{d,t}$ and its confidence $\gamma_{d,t}$ for UCB-1 are calculated by the expectation–maximization (EM) algorithm which maximizes the likelihood of the observed user behavior. This approach was suggested in [31] in combination with ranking by UCB-1.

At the **E-step**, we estimate the hidden variables E_d^t by calculating posterior probabilities of examinations $P(E_d^t = 1|\, UB_t)$, where UB_t is the user behavior observed at step t (see Appendix for details).

Further, we use them at the **M-step**:

$$\gamma_{d,t+1} = \sum_{\tau=1}^{t} P(E_d^\tau = 1|\, UB_t), \quad W_{d,t+1} = \sum_{\tau=1}^{t} C_d^\tau. \quad (4)$$

- Calculation of the **posterior distribution** $p_{d,t}(r)$ for Bayesian bandits relies on the Bayesian inference, see Appendix for details.

Note that parameters $F_t(d)$ updated by the proposed rules aggregate all the user feedback observed on a document d in response to a query q. Hence, ranking by decreasing $S_{B,t}(F_t(d))$ (see Alg. 1) utilizes all the user feedback observed in response to q. In contrast, Radlinski et al. [27, Alg. 2] used a dedicated instance of the SMAB algorithm for each position of the SERP, i.e., took into account only the user feedback on that position while choosing the document for it. While it was appropriate for the problem they solved (increasing the SERP diversity), it may be not the best solution for our case. Indeed, position and previous documents influence only the examination probability of the document d (not r_d) under the examination hypothesis, and our update rules take this influence into account. Hence, aggregation of user feedback on d over all the contexts within the same query seems to be rational. In order to include the approach from [27] in the set of baseline algorithms we experiment with, we adopted it to our problem setup by the following modification of BBRA.

Adoption of [27]. The algorithm stores a dedicated set of parameters $F_{t,i}(d)$ for each position i. The document $d_i(t)$

for the position i is then chosen from the documents which are not chosen for the previous positions, by maximizing $S_{B,t}(F_{t,i}(d))$, and the user feedback on $d_i(t)$ (clicked or not) is used to update only the parameters $F_{t,i}(d_i(t))$.

Adoption of ϵ-greedy. To test the effectiveness of exploration provided by Algorithm 1 with different scoring functions S, we compare them with ϵ-greedy [34], which is the simplest and widely used way to add exploration to any ranking algorithm. We apply it to our exploitative algorithms (see Section 7): for each position i, we choose the top document from the exploitative ranking that was not chosen for the previous positions with probability $1-\epsilon$, and choose a random document from D that was not chosen for the previous positions with probability ϵ.

4. PRIOR DISTRIBUTION PREDICTION

In this section, we discuss setting prior values F_0 of parameters, which is needed for initialization of the BBRA algorithm (see Algorithm 1). Note that we use variables r, W_t, γ_t, F_t, $\{p_t(r)\}_{r\in[0,1]}$ to denote the parameters associated with the pair (q,d) under consideration. While estimation of r is a standard task, prediction of confidence of this estimate is a rather novel problem. Diaz et al. [13, 12] used arm-specific prior estimation of r and set confidence of this estimate to be constant over arms, when considering search verticals as arms. Zhu et al. [37] formulated the problem of confidence prediction in another context and suggested an approach to it when r is defined by a language model based retrieval model. In contrast, our below-described approach assumes that the estimate r is generated by a ranker learned on hundreds of features, which is more realistic for a large-scale search engine, and proposes a set of features useful specifically for such type of prediction.

In our task, without this prediction, the BBRA algorithm would consider all the estimates of r as equally confident and would explore all the documents equally, either placing even clearly irrelevant documents on the top positions or minimizing exploration and so reducing to the standard exploitative ranking. The former case may lead to a short-term but dramatic drop in query performance and irrevocably decrease the market share of the web search engine. Thus, defining documents whose exploration would be the most effective is critical for incorporation of the BBRA algorithm into a web search system. Note that this practical problem was not considered in any related studies on bandit algorithms. For the case of the BBRA algorithm based on Bayesian bandits or UCB-1, we propose the following solution for it.

In the case of Bayesian bandits, we suggest to set $\{p_0(r)\}_{r\in[0,1]}$ to be a beta distribution for each query-document pair, as it was done in [13, 12] for a query-vertical pair, because it brings the following substantial advantages to our framework. First, a beta distribution is determined by only two parameters, α and β. From the viewpoint of our task, it is important to set just two independent principal parameters, estimate of r and its confidence. In terms of distribution, they can be considered as a mean value and a mean absolute deviation from the mean, respectively, which, in the case of the beta distribution, can be expressed through α and β in the following way:

$$\begin{cases} Er = \frac{\alpha}{\alpha+\beta} \\ E|r-E(r)| = \frac{2\alpha^\alpha\beta^\beta}{B(\alpha,\beta)(\alpha+\beta)^{\alpha+\beta+1}} \end{cases} \quad (5)$$

Thus, after estimation of Er and $E|r-E(r)|$ for a particular query-document pair, we are able to set a beta distribution for it by obtaining α and β as solutions of the system of Equations 5 by computational techniques. We use the downhill simplex algorithm [26] here.

Second, when we start with a beta distribution within the negligent Bayesian inference approach (see Equation 2), the posterior distribution $p_t(r)$ we obtain at each step of the inference is also some beta distribution. This allows us to represent the posterior distribution by only two numbers in our memory storage. Moreover, if $p_0(r)$ is uniform on the interval $[0,1]$ in Equation 2, the parameters of the posterior distribution of r are $\alpha = W_t+1, \beta = \gamma_{t+1}-W_t+1$. It means that we can interpret our prior beta distribution as if the arm corresponding to the query-document pair was pulled $(\alpha+\beta-2)$ times with $(\alpha-1)$ of them successfully. Then, it is straightforward to use this distribution also for setting prior values of UCB-1 parameters γ and W according to their definition in Section 3.1.1: $\gamma_0 = \alpha+\beta-2, W_0 = \alpha-1$.

Now, the only open question is how to predict the mean value and the mean absolute deviation for each query-document pair. For this purpose, we use a vector $x = x_{q,d}$ of several hundred ranking features of the production ranker of Yandex[1], a popular search engine used by millions of people from different countries. For prediction of $E(r|x)$, we just relied on a model M_1 based on gradient boosted decision trees (GBDT) [14] and trained by minimizing MSE. The straightforward choice of a ground truth would be to use values of r_d inferred from logs for each query-document pair separately. However, we infer an aggregate value of r_d for all the pairs with the same relevance label manually assigned by professional judges hired by Yandex, since, in our experiments, the user feedback is generated on the basis of these aggregated r_d values (see Section 6 for details).

At the next step, we calculate predictions $M_1(x)$ for examples of a held-out part of the training data and train a new model M_2 to predict absolute errors $|r-M_1(x)|$ of predictions $M_1(x)$ also minimizing MSE. Then, we consider $M_2(x)$ as an estimate of $E(\left|r-E(r|x)\right| |x)$. Note that the signed error $(r-M_1(x))$ of prediction $M_1(x)$ cannot be predicted at any reasonable quality level: if we could do so, we would just improve $M_1(x)$ by correcting it by that prediction of the signed error. On the contrary, the absolute error of prediction turns out to be predictable. In our experiments, M_2 trained on the same features as M_1 outperformed a simple baseline by 19.6% in terms of MSE. This baseline predicts the same constant value for all the query-document pairs, which is equal to the average absolute error over all the pairs from the training set.

However, it is reasonable to expect that our prediction of M_1 error may be significantly strengthened by some features which are not useful for M_1 itself. We consider several ones based on predictions by M_1, which are closely related to their errors. Namely, for a query-document pair (d,q), we use: 1) $M_1(d,q)$; 2) the rank of d according to $M_1(d,q)$ among all the documents for q from the training data set, denote their subset by D_q; 3) $Avg(M_1,q)$ and 4) $StDev(M_1,q)$ are the average and the standard deviation of $M_1(d',q)$ over $d'\in D_q$ respectively; 5) $(M_1-Avg(M_1,q))$; 6) $(M_1-Avg(M_1,q))/StDev(M_1,q)$. Experimental results in Table 1 show that each new feature strengthens M_2 in terms

[1]yandex.com

Table 1: Gain in MSE for new features of M_2, %

Comparison with	1	2	3	4	5	6	All
baseline	6.95	1.83	0.19	1.19	3.57	2.70	8.19
baseline + all others	1.55	0.00	0.02	0.64	0.24	0.00	-

of MSE if added to the baseline vector of features (used for M_1) and each, except for features 2 and 6, does if added to the baseline features supplemented by all other new ones. All the differences, except for zeroes, are significant ($p < 0.05$). Thus, prediction $M_1(q,d)$ and its distribution over all the documents for q make its error prediction much more precise. In further experiments, we use the best of our versions of M_2, i.e., trained on all the baseline and new features.

The key point of both predictions is the choice of the loss function. Now we explain why minimizing merely MSE while training M_1 and M_2 allows us to reach a good estimate of $E(r|x)$ and $E(\left|r - E(r|x)\right| \,|x)$, respectively. It is well-known from statistics that the solution of the optimization problem

$$f^* = \arg\min_f E(h - f(x))^2$$

is $f^*(x) = E(h|x)$. Further, training of GBDT model tends to minimize MSE of prediction on the training data set, i.e., to minimize $E_{data}(h - f(x))^2$ where expectation is taken over the joint distribution of all the values of (h, x) in the training data set. Since our data set is supposed to be a large enough uniform sample from the real joint distribution of (h, x), $E_{data}(h - f(x))^2$ is close to $E(h - f(x))^2$. Therefore, it is reasonable to expect that the trained model provides an unbiased estimate of $E(h|x)$. Finally, we apply this reasoning to M_1 with $h = r$ and to M_2 with $h = \left|r - E(r|x)\right|$. In Section 7, we confirm the positive effect of these predictions on the cumulative query performance by experimental results.

5. INTRODUCTION OF BANDITS TO LTR

A modern ranking system is usually based on hundreds of features incorporated into a scoring function $f(q,d)$ trained to maximize some ranking quality measure. In Sec. 4, we use these features to train models $M_1(q,d), M_2(q,d)$ that predict the mean value of r_d and its deviation from the mean. When we switch from using ranking based on $f(q,d)$ to the ranking method produced by Alg. 1, we may observe a drop of ranking performance for the first several issues of each query. This drop is caused by two different reasons, one is inevitable and acceptable, and another one cannot be accepted.

First, Algorithm 1 allows some exploration whose rate is controlled by the exploration parameter. At the start of running Algorithm 1, this exploration leads to performance decrease that pays off, however, by the increase in quality aggregated at some horizon. This is in line with the main objective of OLREE to reach the best cumulative performance. Another reason of the performance drop is that the exploitative component of Algorithm 1 can be worse than $f(q,d)$. In fact, fully exploitative version of Algorithm 1 ranks documents by the outcomes of $M_1(q,d)$, which is trained via minimization of MSE in a pointwise manner. At the same time, well-known pairwise and listwise learning-to-rank algorithms, which are directly focused on finding the best ranking, are known to often outperform pointwise methods [24] and are widely used in search systems. Therefore, we expect that M_1 has lower ranking quality than $f(q,d)$ in general.

On the other hand, we cannot substitute $M_1(q,d)$ by $f(q,d)$ in Alg. 1, since scores $f(q,d)$ cannot serve as reasonable estimates of r_d in the general case. In fact, the values of $f(q,d)$ can be much greater or much smaller than the values of r_d for a given query q, despite producing the best achievable ranking (from the viewpoint of exploitation). This section is devoted to the problem of incorporating both $f(q,d)$ and $M_1(q,d)$ in a single model $M_{LTR}(q,d)$ that would have both advantages: (1) it should estimate r_d as precisely as possible, (2) it should produce the best possible exploitative ranking.

If we had a model that perfectly predicts r_{d_j}, the outcome scores would also provide the ideal ranking of documents d_j, $j = 1, 2, \ldots, k$. Therefore, when improving the quality of a predictive model, we expect that its ranking quality will be also improved (the intuition of all pointwise ranking models). Our approach is based on the reverse reasoning: when improving ranking quality of a model, we usually expect to increase or, at least, not decrease its predictive quality. Since $f(q,d)$ is supposed to produce the best achievable exploitative ranking, we suggest to correct $\widehat{r}_d = M_1(q,d)$ in such a way that the ranking produced according to corrected estimates $\widetilde{r}_d = M_{LTR}(q,d)$ would coincide with the ranking by $f(q,d)$. At the same time, we tend to minimize deviation of $\widetilde{r}_d$ from $\widehat{r}_d$, as it is regarded as the best possible pointwise estimate of r_d. This formulation naturally leads to the following particular case of the isotonic regression problem with a complete order [5, Section 1].

Assume $d_1, d_2, \ldots, d_n$ is the ranking produced by f, that is, $f(q,d_1) \geq f(q,d_2) \geq \ldots \geq f(q,d_n)$. The task is to find $\widetilde{r}_{d_j} = x_j$, $j = 1, 2, \ldots, n$, such that

$$\begin{cases} x_1 \geq x_2 \geq \ldots \geq x_n \\ \sum\limits_{i=1}^{n} (x_i - \widehat{r}_{d_i})^2 \to \min \end{cases}$$

Applying the necessary Karush-Kuhn-Tucker conditions implies (see [5, Section 2] for details) that the unique solution of this problem has the following form. All the documents are to be divided into consecutive groups $\{d_1, \ldots, d_{k_1}\}, \ldots, \{d_{k_m+1}, \ldots, d_n\}$ to have the identical values $\widetilde{r}_d$ inside each group: $\widetilde{r}_{d_1} = \ldots = \widetilde{r}_{d_{k_1}} = \frac{\widehat{r}_{d_1} + \ldots + \widehat{r}_{d_{k_1}}}{k_1}, \ldots, \widetilde{r}_{d_{k_m+1}} = \ldots = \widetilde{r}_{d_n} = \frac{\widehat{r}_{d_{k_m+1}} + \ldots + \widehat{r}_{d_n}}{n - k_m}$. The optimal partition of the document set into such groups can be found by the Pool Adjacent Violators algorithm [5, Section 3] which has a linear complexity $O(n)$. It starts with regarding each document as a separate group and iteratively merges neighbor groups, if the average of $\widehat{r}_d$ in the upper group is less than in the lower one. In order to keep ranking produced by LTR, we break ties in each group in accordance with the ranking scores $f(q,d)$: $\widetilde{r}_d := \widetilde{r}_d + 0.0001 \cdot f(q,d)$.

While correcting the $\widehat{r}_d$ values, we keep their confidence constant for simplicity. Thus, we have the following
Procedure of LTR-correction of $F_0(d)$:

- transform $\widehat{r}_d$ to $\widetilde{r}_d$ by isotonic regression with breaking ties,
- keep the customary γ_d.

Our experiments show that, as we expected, this correction increases performance not only in terms of NDCG@10 up to the level of the production baseline LTR currently used in Yandex (by 0.79%), but also in terms of MSE by 1.1%.

There are several ways to make use of this scheme in practice. First, we can combine any LTR ranking with our prior distributions of r_d before the start of the BBRA. Then the BBRA starts from the performance of LTR ranking and further improves it in accordance with the collected information. Second, it is possible to periodically update the user

behavior features participating in the LTR model by making use of clicks gathered by BBRA and correct the current posterior distribution in accordance with the updated LTR ranking. We can expect that this periodical correction will strengthen our approach, because the LTR algorithm could exploit user feedback more effectively than the BBRA, which, on the other hand, defines the rate of exploration needed for improving LTR. In particular, it may use query-document features to get information for other query-document pairs from observations for a particular pair, i.e., propagate this information over documents or queries. Third, it is also possible to re-train the LTR algorithm periodically on the ground of the fresh values of user behavior features. Because of the large number of other settings varying in our experiments, we implement only the first idea and show that it gives us a remarkable advantage over both LTR and BBRA working separately.

6. EXPERIMENTAL SETTINGS

Here, we describe our experimental setup to evaluate and compare different OLREE algorithms described in Section 3.

6.1 Data set

We collected the log of the live stream of search queries submitted to Yandex during two weeks of November, 2013 (referred as *Logs* below). We randomly sampled 0.003% of query issues from it. Since our algorithm can hardly explore r_d for very rare queries and hence these queries may introduce noise into our analysis, we filtered out the queries whose frequency estimated on *Logs* is less than 3 issues per day on average. As a result, we obtained a data set of 28 746 query issues, which represent 37.5% of the search traffic. For each query q from the data set, we united sets of top-10 results provided by different production rankers (we also use the best of them further and refer to it as *LTR* below) for this query to obtain the final set D_q of documents to assess. All the documents from D_q were assessed for relevance using the 5-grade scale (Perfect, Excellent, Good, Fair, Bad). In Section 7, we experiment with sets of top-ranked documents D (see Algorithm 1) of different predefined sizes m (from 5 to 30), which can exceed D_q for some queries q. Since, in reality, we always have as many documents to experiment with as we need, in such cases we added several documents to D_q (to make its size equal to m) by sampling them from all the Bad documents in the data set and labeling them as Bad with respect to the query q.

Further, we randomly split all the queries from the data set into several parts: $Train_1$ (25% of all the queries), $Train_2$ (25%), $Train_3$ (20%), and $Test$ (30%). We used the sets $Train_1$ and $Train_2$ to train the predictors M_1 and M_2, introduced in Section 4, correspondingly. As we described in Section 3, OLREE algorithms depend on a few parameters defining the exploration rate. Another parameter influencing the exploration rate is the number $|D|$ of the top documents the bandit algorithm works with. We used $Train_3$ to find their optimal values for each algorithm by exhaustive search in the parameter space. Finally, we tested M_1, M_2, LTR-correction and different OLREE algorithms on $Test$.

6.2 Realistic simulation of production

Since it is very risky for the search engine's market share to experiment with exploratory algorithms on real users, most of related studies [17, 27, 30] performed only experiments with simulated user feedback. In our experiments, we try to simulate a production environment much more realistically than they did. First, in contrast to [27, 30], we use not artificial but real queries and documents. It is especially important for our approach, because, for each query-document pair, we need not only its value of r_d but also its vector of features.

Second, we are the first to take into account different query frequencies while evaluating an effect of OLREE algorithms on the aggregated quality of the web search system during a fixed period by means of simulations, whereas all the previous papers used simulations of the equal number of query issues for each query. It is well-known that the more steps a bandit algorithm is able to make, the more effective it is in comparison with an exploitative strategy in terms of the cumulative reward. Therefore, for appropriate evaluations, it is required to simulate realistic query frequencies. We perform this as follows.

Let us consider a query issue i from $Test$ referring to a query q whose frequency estimated on Logs is I_q issues per 10 days. Naturally, we simulate I_q consequent submissions of q, run a tested algorithm on them. We repeat this simulation 5 times and consider the average query performance (see Section 6.3 for the description of our metrics) over all these issues as an estimate of the query performance of i, since i represents a random sample from these issues in $Test$. Then, to evaluate the overall performance of the algorithm, we average the obtained results over all the query issues from $Test$. Besides this cumulative performance, we also evaluate the *final* quality the algorithm is able to provide after 10 days by measuring the quality of ranking produced by the exploitative version of the algorithm (see Section 7) on the basis of all information it obtained during I_q issues of q and averaging this value over all the query issues from $Test$.

Third, for each simulated query issue, the tested algorithm provides the list of $k = 10$ documents, and we simulate the user feedback on them by DCM (see Section 3). For this purpose, we assume $\lambda_i = \lambda$ in Equation 3 and assign the same r_d values to all the query-document pairs with the same relevance judgments, as it was done in [17, 31]. However, unlike in these studies, we do not set λ and r_d to extreme ad-hoc values, but infer more realistic estimates by maximizing the likelihood of the user feedback on queries from $Train_1$ + $Train_2$ stored in *Logs* under the DCM model adapted in the following way. In terms of Equation 3, we substitute r_d, which is attributed to each individual document d, with r_j (where $j \in \{1, 2, \ldots, 5\}$ identifies the relevance label of d), as it was done in [9]. We also use the standard way to identify satisfied clicks on logs [3, 4, 10]. The inferred values r_j, $j = 1, 2, \ldots, 5$, are also used as the ground truth for training M_1 and M_2 (see Section 4).

Several studies [23, 32, 21] also suggested an alternative approach to offline experiments with bandit algorithms, which is called the replay method and is based on utilizing logs of the user-system interaction. However, it was applied only to bandit algorithms choosing one arm at each step (e.g., by recommending one news article) and is effective only when each arm was chosen by the logging policy a sufficient number of times. The first problem we would face when trying to apply this method to our task is that our OLREE algorithms regularly suggest the permutations of top-10 documents which are absent in the logs at all. Indeed, even just 10 documents can be ranked in $10! = 3628800$ ways. Collecting a "random

bucket" (as it was done in [18, 23, 21]) for such a number of permutations will be possible only for a tiny set of extremely popular queries, will take lots of time and will leave lots of users dissatisfied with low-quality rankings. Second, one of our goals is not only to find the best permutation of a given top-10, but also to increase its overall relevance by introducing more relevant but currently lower ranked documents into it. It means that the replay method would be infeasible even for popular queries.

6.3 Metrics

In our evaluations, we use the following two measures of query performance. First, we consider the widely used NDCG measure. For this purpose, we assign an NDCG-wise relevance gain $\frac{2^g-1}{2^4}$ [8] to each query-document pair, where $g \in \{0,1,2,3,4\}$ corresponds to its relevance judgment (from Bad to Perfect). Because of the proprietary nature of the *LTR* algorithm, we do not report absolute values of measures, but instead report absolute changes in measure value with respect to the LTR ranking in percentages: $\Delta NDCG = (NDCG@10 - NDCG@10_{LTR}) \cdot 100\%$. Second, we are interested in the number of satisfied clicks on the top 10 positions, since this metric is the underlying objective of our bandit ranking approach described in Section 2. We present it in the form of *regret* which is the common metric (to be minimized) in the SMAB problem. In our case, it equals the difference between the cumulative reward of the ranking by true r_d values and the cumulative reward of the tested algorithm. After averaging regret over all queries, we measure its relative change: $\Delta Reg = \frac{regret - regret_{LTR}}{regret_{LTR}} \cdot 100\%$.

Since the optimal exploration rate of a bandit-based algorithm strictly depends on the number of steps which is defined by the query frequency in our experiments, we split $Test$ into 4 frequency groups of the same size in order to tune exploration rate and to compare different algorithms separately on each group. The corresponding frequency intervals (number of issues per 10 days) are $[30, 150), [150, 750), [750, 5500), [5500, 1.5 \cdot 10^6)$ for Groups 1-4 respectively. To define a frequency group in practice, one can utilize methods of query frequency prediction, e.g., on the basis of historical data [29].

7. EXPERIMENTAL RESULTS

We first describe different methods we experimented with. Each one consists of the following basic components with options, which are denoted as follows:

- OLREE algorithm (see Section 3): U (MeanUCB-1 by default, UCB-1 is indicated by the postfix *orig*) or B (MeanBayes), which include scoring function $S_t(d)$ and update rule; additional options (see Section 3.3): *10* means that a dedicated instance per each position is used (adoption of [27]), e stands for an exploitative version ($\alpha = 0$ in Equation 1 for UCB-1, MeanUCB-1 and MeanBayes, $\alpha_{low} = \alpha_{up} = 0.5$ for Bayesian bandits), ϵ denotes ϵ-greedy applied to the exploitative version, n means that the negligent inference approach is used (honest approach is applied by default);
- prediction of prior r and γ (see Section 4): the best constant (over all query-document pairs) estimate of r and $\gamma = 1$ (setting from [31]) are used by default, R stands for prediction of r by M_1 and the best constant estimate of γ, prediction of r by M_1 and prediction of γ by M_2 are denoted by P;
- correction of prior estimates by LTR (see Section 5) is marked by L and is not used by default. Each method is denoted by the combination of its options.

Besides the production LTR ranking, we consider the following two baselines, which do not take into account prior information on relevance the search system possess: U_orig is our implementation of the method suggested in [31], U10_orig is our adoption of Algorithm 2 from [27] to our task.

We observed that, for any query with more than 10 000 submissions within a simulated 10-day period (82% of Group 4), each method with optimal settings (see details below) is able to provide nearly optimal ranking after 10 000 submissions if switches to exploitation. Therefore, in response to all further issues of each such query, we showed the constant document list that is provided by the exploitative version of the method under consideration on the basis of all information it obtained during the first 10 000 issues. Note that the related studies [27, 14] modeled 1000 steps of bandit algorithms at maximum.

Although each SMAB algorithm we adopted to the OLREE problem has an *exploration parameter* that controls the exploration rate (α for UCB-1, MeanUCB-1 and MeanBayes, $(\alpha_{low}, \alpha_{up})$ for Bayesian bandits and ϵ for ϵ-greedy), it may be not sufficient to find the optimal strategy for the OLREE problem, which requires to choose many documents at each step. Thus, it may be useful to manually restrict the number of documents $m = |D|$ the OLREE algorithm explores. We tune $m \in \{5, 10, 15, 20, 25, 30\}$ and the exploration parameter for each method to be tested and for each frequency group on $Train_3$ individually. In the case of $m = 5$, we place documents assigned to positions 6-10 by the LTR ranking on the same positions on the SERP at each query issue.

As Figure 1 and Table 2 show, expectedly, the optimal m increases with query frequency and with complexity of the method. The optimal m for the best methods is 15 or higher, what confirms that the exploration of documents normally placed out of the first page has a high potential for ranking improvement. The optimal values of α range from 0.01 to 0.44 for different MeanUCB-1 methods and from 0 to 1.32 for MeanBayes methods.

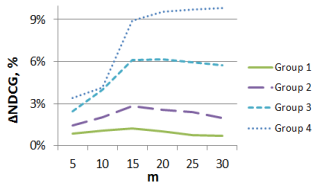

Figure 1: Setting m for UPL

Method	Frequency groups			
	1	2	3	4
U10_orig	5	5	5	15
U_orig, U	5	5	15	20
UR	5	10	15	25
UP	10	15	15	25
UPL	15	15	15	30
UePL	15	15	15	25
UεPL	0	0	0	15
UnPL	5	5	10	10
BR	5	5	15	20
BP	5	10	15	30
BPL	15	15	15	30

Table 2: Optimal m

We conducted experiments with all the methods on $Test$ according to the procedure described in Section 6 and used Student's paired t-test to compare the obtained results. Each of the methods based on MeanUCB-1 or MeanBayes outperformed its analog, where UCB-1 (Bayesian Bandits correspondingly) was used and the other options were the same. Therefore, we present results only for MeanUCB-1-based and MeanBayes-based methods in comparison with the baselines U10_orig and U_orig. The results observed in the experiments with the methods based on UCB-1 and Bayesian bandits are analogous. The performance on $Test$ in terms of cumulative ΔNDCG (for each frequency group and aggregately), final ΔNDCG, and ΔReg (see Section 6.3) is reported in Table 3. First, we see that among two baselines $U10\_orig$ and U_orig, which do not use any prior information, U_orig performs significantly better according to each column of the table

Table 3: Performance evaluation of the previous bandit algorithms and our methods based on priors

Method	ΔNDCG,%						ΔReg,%
	cumulative					final	
	G.1	G.2	G.3	G.4	All	All	All
U10_orig	-1.77	0.05	1.22	6.88	2.26	4.75	-17.3
U_orig[31]	-0.40	1.16	3.87	9.40	4.21	7.59	-25.0
U	-0.36	1.17	4.07	9.40	4.27	7.64	-24.8
UR	0.01	1.27	4.23	9.66	4.55	7.76	-25.2
UP	0.34	1.99	5.57	10.13	5.25	8.51	-29.8
UPL	0.93	**2.98**	**6.12**	**10.35**	**5.78**	**8.91**	**-31.5**
UePL	0.84	2.84	5.61	9.16	5.21	7.98	-25.9
UϵPL	0.84	2.84	5.61	9.64	5.37	8.31	-28.3
UnPL	0.65	1.25	1.46	2.70	2.04	3.00	-0.9
BR	-0.08	1.35	3.97	9.21	4.33	7.50	-21.6
BP	0.38	1.88	5.42	9.38	4.94	8.03	-27.0
BPL	**0.94**	2.98	6.05	9.72	5.53	8.53	-30.4

($p < 0.01$). Being based on an estimate of the mean, U is only slightly better than U_orig.

Second, we observe that our approach to utilizing prior information from the current production system allows to remarkably increase the cumulative performance. The method UR using prediction of r_d significantly outperforms the former three methods in terms of ΔNDCG according to each column ($p < 0.05$). The prediction of γ by M_2 used in UP and BP remarkably strengthens ($p < 0.01$) each of the methods UR and BR, and, then, the LTR-correction provides the additional notable increase in quality ($p < 0.01$ for UPL vs UP, BPL vs BP).

Now we examine how the deterioration of each component of the best method (UPL) affects its performance. The exploitative version UePL is significantly weaker ($p < 0.01$) than UPL and BPL, except for Group 1. Within Group 1, the number of submissions of a query is too small to significantly gain from exploration. Next, random exploration by ϵ-greedy (UϵPL) turned out to be effective only for Group 4 and with low exploration rate (optimal $\epsilon = 0.04$), because of the high cost of each exploratory choice (a high risk of choosing irrelevant document), and does not still reach the performance of UPL or BPL. Finally, the negligent inference approach (UnPL) performs very poorly (especially for frequent queries) due to the low learning rate. This emphasizes the importance of applying an appropriate model of user behavior to infer as much information from logs as possible.

Figure 2 presents the dynamics of the cumulative ΔNDCG on Group 1 (left) and Group 4 (right) for the principally different methods: the best exploitative one (UePL), the best method with the negligent inference (UnPL), and the best methods with the different prior settings (U, UR, UP, UPL). We see that prior information plays a critical role for queries with low frequency, and learning rate defined by an OLREE algorithm becomes much more important for more frequent queries. However, given the OLREE algorithm, the prior settings notably influence its performance even for Group 4.

Another aspect of search quality, which cannot be directly measured by the cumulative NDCG, is the frequency of significant drops in query performance in comparison with other search engines. In the case of the method U, 14.4% and 5.1% of query issues during the first day have ΔNDCG less than −10% and −20% respectively, while these shares are only 2.4% and 0.33% for UPL. It means that, in the case of U, much more users would be negatively affected by the severe degradations of the quality and could completely switch to another search engine.

Finally, the analysis of UPL performance for individual queries confirms a clear idea: the gain of proper OLREE

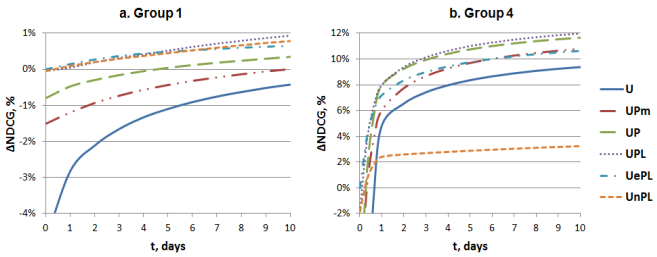

Figure 2: Dynamics of cumulative ΔNDCG over t days (at 1st step for $t = 0$)

algorithms over the LTR ranking they start with depends on the quality of the latter. Figure 3 presents the cumulative ΔNDCG of UPL averaged over all queries of a known frequency whose initial LTR performance lies within a given range. Naturally, it crucially decreases with the growth of LTR quality, because chances of each exploration action to be successful (i.e., to find a new relevant document) decrease. It means that taking current ranking quality into account while setting the exploration rate for the query has a high potential to increase performance of OLREE algorithms. In practice, methods of query performance prediction [16, 36] can be used for this purpose.

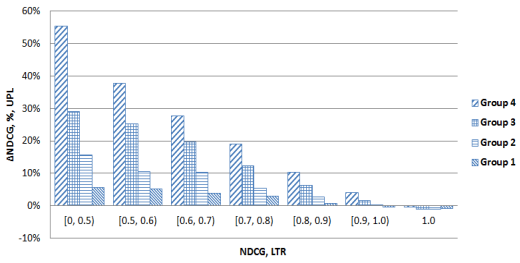

Figure 3: Dependence of ΔNDCG of UPL on LTR quality

8. RELATED WORK

Almost all the attempts to represent the OLREE task as the SMAB problem (exceptions are discussed below) could be divided into the following three types, each giving its own definition for a bandit problem, an arm, its trial and its reward. The first two types rely on the contextual bandit approach and regard submissions of all the queries as an entire bandit problem. The first one considers each submission as a trial and a linear ranking algorithm parametrized by a weight vector ω as an arm. The reward is then defined by the whole user behavior on the SERP (for example, it can be opposite to abandonment). As long as arms here are vectors, these approaches [35, 28] apply boosting rather than standard bandit algorithms to find an optimal value of ω. The second interpretation [22, 25] considers a query-document as an arm, the fact of an arm trial consists in two conditions: the corresponding document was presented on SERP and was examined by the user, and the reward is a click (reward=1) or a skip (reward=0). By assuming that the click probability linearly depends on a number of features of the query-document pair, one is able to estimate this probability and confidence in this estimate for each pair. One major problem of these two approaches is that linear rankers are known to be suboptimal with respect to such state-of-the-art LTR approaches as neural networks [6] and decision trees [7]. Another major weakness of these approaches is their low

degree of freedom, which does not allow to derive the best possible ranking for each individual query.

Within the third type of approaches, a separate bandit problem setup corresponds to issues of each query (or even more locally, see description of [27]). Definitions of an arm and a trial are the same as in the second approach. While considering each query separately and each document as an arm, these approaches allow to reach the best ranking for each of the relatively frequent queries individually. If we assume that the first result is always examined and the lower results are never examined, as it was made in [22], this approach reduces to the ordinary SMAB formulation: the document on the first position is the chosen arm. However, in web search ranking, user satisfaction significantly depends on the lower documents, thus maximizing relevance of only the first document is suboptimal.

A more appropriate method was proposed in [27] for the problem of web search diversification. There was considered a dedicated instance of bandit algorithm for each position within issues of a fixed query. As we discussed in Section 3.3 and showed experimentally in Section 7, our method is more effective in our framework.

Sloan and Wang [31] proposed to use one bandit algorithm instance for one query, which accounts for all clicks on different positions. However, they could not obtain any profit from exploration in terms of NDCG. In this paper, we significantly strengthen their approach by incorporating it into the production ranking and by utilizing prior information.

There were also different attempts to reduce the OLREE problem to the SMAB problem [30, 33, 1], which do not formally correspond to the above described classes of approaches. The work of Slivkins et al. [30] belongs to the second type of approaches, excluding the linear model assumption, and considers a "perfect world" scenario: given a query, all the documents are assumed to be represented by points of a metric space with such a metric $D(x, y)$ that the probability r_d of a click on a document d satisfies the following inequality: $|r_{d_1} - r_{d_2}| \leq D(d_1, d_2)$. It allows to develop and theoretically justify a contextual bandit algorithm which propagates inferred information from one document to the other ones positioned close to this document in the metric space. However, the practical realization of such a space construction remains an open question. In fact, it requires an accurate estimation of differences between click probabilities of each two documents. It is not even clear if this problem is simpler than ranking of documents by their click probability, the problem which we actually solve.

9. CONCLUSIONS

In this paper, we investigate a new formulation of the exploration–exploitation dilemma in online learning to rank (OLREE) in terms of the SMAB problem. It represents a general approach to applying a variety of SMAB algorithms to the OLREE problem. Further, previous methods proposed in the literature for this problem did not consider the following critical practical issue: any commercial search engine is based on a highly optimized ranking algorithm which utilizes hundreds of features and can not be just substituted with a principally new one. Thus, an OLREE algorithm should be "married" with the current production ranker and handle all its advantages. To address this issue, we developed a general framework for introducing our bandit-based learning method into any default ranking system. Finally, we applied the whole scheme to several SMAB algorithms and experimentally demonstrated that it enables to notably increase the performance of a major search system in terms of NDCG measure averaged over a 10 day period.

We plan to extend this work by making our OLREE algorithm more contextual, i.e., able to effectively aggregate user feedback over different contexts to produce the optimal (in OLREE terms) ranking in the current context at each step.

Appendix

We give here some details about the update rules described in Section 3.3. We introduce random events $A_l = \{C^t_{d_j} = 0 \text{ for } j = l+1 \text{ to } k\}$ that there was no click below position l at query issue t. We also denote the lowest click position observed at query issue t by $l(t)$ (to be considered not as a random variable, but as a fixed value determined by the observed user behavior after the query issue t). In the equations below we assume all the probabilities under the condition $C^t_{d_{l(t)}(t)} = 1$ and under the condition that the current estimates of r_d (point estimate $\widehat{r}_d$ or $\overline{r}_d$ in the cases of UCB-1, MeanUCB-1 and Bayes-UCB, posterior distribution $p_{d,t}(r)$ in the case of Bayesian bandits) are true values for all the documents. For brevity, we omit these conditions and use d_j instead of $d_j(t)$ below. Now, we describe update rules specific for UCB-1 and Bayesian bandits:

- In the case of UCB-1, E-step is the following. For $i = 1, \ldots, l(t)$, we obviously put $P(E^t_{d_i} = 1 | UB_t) = 1$. For $i > l(t)$, we obtain:

$$
\begin{aligned}
P(E^t_{d_i} = 1 | UB_t) &= P(E^t_{d_i} = 1 | A_{l(t)}) = \\
&= P(\bigcap_{j>l(t)} \{E^t_{d_j} = 1\} | A_{l(t)}) = \\
&= \frac{P\left(A_{l(t)} \cap \bigcap\limits_{j>l(t)} \{E^t_{d_j} = 1\}\right)}{P\left(A_{l(t)} \cap \bigcap\limits_{j>l(t)} \{E^t_{d_j} = 1\}\right) + P\left(A_{l(t)} \cap \bigcap\limits_{j>l(t)} \{E^t_{d_j} = 0\}\right)} = \\
&= \frac{P(E^t_{d_{l(t)+1}} = 1) \cdot \prod\limits_{j>l(t)} P(C^t_{d_j} = 0 | E^t_{d_j} = 1)}{P(E^t_{d_{l(t)+1}} = 1) \cdot \prod\limits_{j>l(t)} P(C^t_{d_j} = 0 | E^t_{d_j} = 1) + P(E^t_{d_{l(t)+1}} = 0)} = \\
&= \frac{\lambda_{l(t)} \cdot \prod\limits_{j>l(t)} (1 - \widehat{r}_{d_j, t-1})}{\lambda_{l(t)} \cdot \prod\limits_{j>l(t)} (1 - \widehat{r}_{d_j, t-1}) + (1 - \lambda_{l(t)})}. \quad (6)
\end{aligned}
$$

The first equality is valid because of the DCM assumptions (3): given that the last click is on position $l(t)$, document $d_{l(t)+1}$ was examined if and only if all the documents below $l(t)$ were examined. Otherwise, all the documents below $l(t)$ were not examined. We use it in the second equality in denominator.

This estimate is similar to the Equations 9 and 10 from [31] (S_i is equivalent to our E_{d_i}). However, authors use only observation of a click or its absence on the document d to estimate E_{d_i} while we utilize information about all the clicks on the SERP. As a result, we have more precise estimates for E_{d_i} under DCM assumptions. This difference is especially remarkable for documents above the lowest click: we exactly know that they were examined.

- In the case of Bayesian bandits, the update rule could be obtained in the similar way by applying the Bayesian inference.

10. REFERENCES

[1] A. Agarwal, D. Hsu, S. Kale, J. Langford, L. Li, and R. E. Schapire. Taming the monster: A fast and simple algorithm for contextual bandits. *arXiv preprint arXiv:1402.0555*, 2014.

[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] P. N. Bennett, F. Radlinski, R. W. White, and E. Yilmaz. Inferring and using location metadata to personalize web search. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 135–144, New York, NY, USA, 2011. ACM.

[4] P. N. Bennett, R. W. White, W. Chu, S. T. Dumais, P. Bailey, F. Borisyuk, and X. Cui. Modeling the impact of short- and long-term behavior on search personalization. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 185–194, New York, NY, USA, 2012. ACM.

[5] M. Best and N. Chakravarti. Active set algorithms for isotonic regression; a unifying framework. *Mathematical Programming*, 47(1-3):425–439, 1990.

[6] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. ICML'05, 2005.

[7] O. Chapelle, Y. Chang, and T.-Y. Liu. The yahoo! learning to rank challenge. *http://learningtorankchallenge.yahoo.com*, 2010.

[8] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. CIKM '09, pages 621–630, New York, NY, USA, 2009. ACM.

[9] A. Chuklin, P. Serdyukov, and M. De Rijke. Click model-based information retrieval metrics. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 493–502. ACM, 2013.

[10] K. Collins-Thompson, P. N. Bennett, R. W. White, S. de la Chica, and D. Sontag. Personalizing web search results by reading level. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 403–412, New York, NY, USA, 2011. ACM.

[11] 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, New York, NY, USA, 2008. ACM.

[12] F. Diaz. Integration of news content into web results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 182–191. ACM, 2009.

[13] F. Diaz and J. Arguello. Adaptation of offline vertical selection predictions in the presence of user feedback. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 323–330. ACM, 2009.

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

[15] F. Guo, C. Liu, and Y. M. Wang. Efficient multiple-click models in web search. WSDM '09, pages 124–131, New York, NY, USA, 2009. ACM.

[16] B. He and I. Ounis. Query performance prediction. *Information Systems*, 31(7):585–594, 2006.

[17] K. Hofmann, S. Whiteson, and M. Rijke. Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. *Information Retrieval*, 16(1):63–90, 2013.

[18] L. Jie, S. Lamkhede, R. Sapra, E. Hsu, H. Song, and Y. Chang. A unified search federation system based on online user feedback. KDD '13, pages 1195–1203, New York, NY, USA, 2013. ACM.

[19] E. Kaufmann, O. Cappe, and A. Garivier. On bayesian upper confidence bounds for bandit problems. In N. D. Lawrence and M. A. Girolami, editors, *AISTATS-12*, volume 22, pages 592–600, 2012.

[20] E. Kaufmann, N. Korda, and R. Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In *Algorithmic Learning Theory*, volume 7568 of *Lecture Notes in Computer Science*, pages 199–213. Springer Berlin Heidelberg, 2012.

[21] L. Li, S. Chen, J. Kleban, and A. Gupta. Counterfactual estimation and optimization of click metrics for search engines. *CoRR*, abs/1403.1891, 2014.

[22] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. WWW '10, pages 661–670, New York, NY, USA, 2010. ACM.

[23] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 297–306, New York, NY, USA, 2011. ACM.

[24] T.-Y. Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, Mar. 2009.

[25] T. Moon, W. Chu, L. Li, Z. Zheng, and Y. Chang. An online learning framework for refining recency search results with user click feedback. *ACM Trans. Inf. Syst.*, 30(4):20:1–20:28, Nov. 2012.

[26] J. A. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[27] F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. ICML '08, pages 784–791, New York, NY, USA, 2008. ACM.

[28] K. Raman, T. Joachims, P. Shivaswamy, and T. Schnabel. Stable coactive learning via perturbation. *ICML*, 2013.

[29] M. Shokouhi and K. Radinsky. Time-sensitive query auto-completion. SIGIR '12, pages 601–610, New York, NY, USA, 2012. ACM.

[30] A. Slivkins, F. Radlinski, and S. Gollapudi. Ranked bandits in metric spaces: Learning diverse rankings over large document collections. *J. Mach. Learn. Res.*, 14(1):399–436, Feb. 2013.

[31] M. Sloan and J. Wang. Iterative expectation for multi period information retrieval. In *WSDM Workshop on Web Search Click Data*, 2013.

[32] L. Tang, R. Rosales, A. Singh, and D. Agarwal. Automatic ad format selection via contextual bandits. In *Proceedings of the 22nd ACM international conference on Conference on information and knowledge management*, pages 1587–1594. ACM, 2013.

[33] H. P. Vanchinathan, I. Nikolic, F. De Bona, and A. Krause. Explore-exploit in top-n recommender systems via gaussian processes. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 225–232. ACM, 2014.

[34] C. J. C. H. Watkins. *Learning from delayed rewards.* PhD thesis, University of Cambridge, 1989.

[35] Y. Yue and T. Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. ICML '09, pages 1201–1208, New York, NY, USA, 2009. ACM.

[36] Y. Zhou and W. B. Croft. Query performance prediction in web search environments. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 543–550. ACM, 2007.

[37] J. Zhu, J. Wang, I. J. Cox, and M. J. Taylor. Risky business: Modeling and exploiting uncertainty in information retrieval. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 99–106, New York, NY, USA, 2009. ACM.