Efficient Approximate Thompson Sampling for Search Query Recommendation

Chu-Cheng Hsieh
eBay, Inc.
2065 Hamilton Ave.
San Jose, CA 95125, USA
chsieh@ebay.com

James Neufeld
University of Alberta, Edmonton
4732 Bolter Hall
AB T6G 2E8, Canada
jneufeld@ualberta.ca

Tracy King
eBay, Inc.
2065 Hamilton Ave.
San Jose, CA 95125, USA
tracyking@ebay.com

Junghoo Cho
University of California, Los Angeles
4732 Bolter Hall
Los Angeles, CA 90095, USA
cho@cs.ucla.edu

ABSTRACT

Query suggestions have been a valuable feature for e-commerce sites in helping shoppers refine their search intent. In this paper, we develop an algorithm that helps e-commerce sites like eBay mingle the output of different recommendation algorithms. Our algorithm is based on "Thompson Sampling" — a technique designed for solving multi-arm bandit problems where the best results are not known in advance but instead are tried out to gather feedback. Our approach is to treat query suggestions as a competition among data resources: we have many query suggestion candidates competing for limited space on the search results page. An "arm" is played when a query suggestion candidate is chosen for display, and our goal is to maximize the expected reward (user clicks on a suggestion). Our experiments have shown promising results in using the click-based user feedback to drive success by enhancing the quality of query suggestions.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Thompson sampling, query suggestions, multi-armed bandit

1. INTRODUCTION

Most major web search applications generate an alternative set of suggested search queries, a.k.a. *related searches*, after an initial search request to help users refine their search intent. Such query suggestions are a natural way to facilitate user engagement in search applications because they are easily understood by users and make use of the existing search infrastructure. Effective query suggestions may take various forms. For example, they may aid users in refining queries to better match their original intent, often by providing alternative wording or adding disambiguating terms. Alternatively, queries may be used as a recommender system to suggest different, but related, concepts or products which might also interest the user. For example, in the e-commerce setting the suggestion "`iphone cases`" for the query "`iphone`" may increase the odds of selling multiple items together.

Because web search settings are typically paired with an abundance of available data, such as indexed search results or user interactions, it is possible to generate query suggestions automatically from data. Often, there are abundant candidates generated from such data and therefore we need a ranking strategy to determine which suggestions to display. However, the selection is difficult because of the dynamic nature of the context. For example, we sometimes favor a refinement suggestion "`iPhone 6`" for a broad intention query "`iPhone`", while we may favor a combined sale suggestion, say "`iPhone 6 case`" for the query "`iPhone 6`".

In this paper, we treat query suggestions as a competition among data resources: we have many query suggestion candidates competing for limited space on the search results page. We demonstrate that multi-armed bandit (MAB) algorithms are particularly well suited to the task of query suggestion. This is due to the fact that the query suggestion problem requires less complex machine learning infrastructure than other tasks. In display advertising, for instance, engagement rates often vary widely across users, and ads are frequently changed, added, or removed by advertisers.[1] As a result, it becomes necessary for the practitioner to predict the engagement rate on an ad using *features* of the advertisement, user, and/or query (a.k.a. the context) with the hope that this learned classifier will *generalize* to novel, unobserved contexts. In the query suggestion setting, how-

SAC'15 April 13-17, 2015, Salamanca, Spain
ACM 978-1-4503-3196-8/15/04 $15.00.
http://dx.doi.org/10.1145/2695664.2695748

[1]Similar observations can be made for recommending news, music, books, etc.

ever, new suggestions are observed less frequently, i.e. at the rate that new products and terms are invented, and suggestion engagement does not vary considerably across users. Consequently, the relevant context can be reduced to a set of query-suggestion pairs. Then, the engagement rate for each pair may be approximated independently.[2] This greatly simplifies the problem as we can avoid designing features and experimenting with different supervised-learning approaches and, instead, directly deploy standard *stochastic multi-armed bandit* techniques, for which simple, computationally efficient, and near-optimal allocation strategies are well understood [7, 2, 12].

Specifically, we propose a lightweight and efficient algorithm that closely resembles the Thompson sampling approach for the so-called Bernoulli-bandit problem. Thompson sampling is a Bayesian approach which has been shown to work exceedingly well in published empirical studies [9, 19] as well having strong theoretical guarantees [12, 1, 8]. The differences between our proposed algorithm and the standard approach result from simplifications necessary to accommodate multiple (M) query suggestions per request, which effectively creates a combinatorially large number of choices per query. In particular, we make the simplifying assumption that the engagement rate on a specific search suggestion is independent of the other $M-1$ suggestions present. While this assumption may not always hold, given the potential dependence between similar suggestions i.e. "`ps3`" vs. "`playstation 3`", we find that it holds frequently enough given the considerable computational savings.

We evaluate the performance of our proposed approach in an offline setting using real user search traffic on `ebay.com` collected using an expanded set of presented query suggestions. This allows us to validate whether our approach, when given a smaller number of suggestion "slots", recovers the optimal suggestions. While this analysis is preliminary, the results are promising and provide sufficient evidence to justify a live production experiment, as part of our future work, where evaluation is considerably more straightforward.

2. RELATED WORK

Data-driven technologies for generating query suggestions are popular in the literature, with approaches differing primarily as to which data sources are used. For instance, as is the case with search ranking methods, many of the query suggestion approaches are purely *document based*; that is, they rely on the (indexed) document corpus to produce suggestions. Popular approaches of this type include clustering similar queries based on the proportion of overlapping search results [4], relating queries using a graph-walk distance over the *query-document* transition graph [5], or directly substituting individual keywords determined to be similar in the text corpus [11]. More recently, query suggestion techniques have leveraged previous user session behaviour, which exploits the hypothesis that users reformulate queries that return poor results, as well as follow up queries with related queries. There are a number of ways in which the resulting query co-occurrence data may then be used to construct suggestions. Notably, Huang et al. [10] were the first to suggest ranking suggestions by observed 1-step query-to-query transition probabilities, Boldi et al. [6] recommend ranking according to a multi-step graph-walk distance metric, and Ozertem et al. [17] propose ranking according to a supervised learning model trained on observed transitions.

Importantly, despite the popularity of these user session based approaches both in academic and industrial settings, they often ignore two critical aspects of the data. First, which of the suggested search queries was clicked by the user, if any; second, the bias introduced by presenting suggestions to the user. For example, an approach that ranks suggestions according co-occurrence rates, as suggested by Huang et al., will bias toward favouring existing query suggestions. This bias creates a positive feedback cycle between the algorithm and the generated data and, consequently, it is difficult for the algorithm to remove a poor suggestion from the list of suggestions or to adapt to changing user preferences. In the machine learning literature this problem is typically referred to as the *exploration-exploitation* dilemma. Stated simply, a suggestion algorithm that learns from past data must trade off the benefits of collecting data on suggestions that are relatively unseen but potentially valuable (exploration), with the benefits of presenting the user with the historically best performing suggestion (exploitation).

Online learning algorithms, sometimes referred to as *multi-armed bandit* methods, are a popular way of addressing this tradeoff (see [7, 2]). These approaches generally manage exploration through modelling the uncertainty surrounding each of the learned predictions. This modeled uncertainty allows the algorithm to ensure that the optimal action is not avoided indefinitely due to a poor (unlucky) estimate of its payoff. Many existing multi-armed bandit approaches come with strong theoretical guarantees and are effective in practical settings, e.g. display advertising [15, 9], news article recommendation [14], and web search ranking [18].

3. MULTI-ARMED BANDITS

The multi-armed bandit (MAB) problem is a sequential allocation task where the learning agent must decide, at each time step, which action to allocate a unit resource to maximize its long term payoff [7]. The phrase *multi-armed bandit* is a reference to the colloquial term for a casino slot machine, which serves as an example for the problem setting: consider the agent to be a gambler who must decide which casino slot machine to play at any given time (out of a set of $K \geq 2$) in order to maximize his expected total winnings. Given that the payout distribution is unknown to the gambler, the gambler must weigh the benefits of playing the best performing machine based on past experience (exploitation) versus trying other machines to gain more understanding of the underlying distributions (exploration).

3.1 Stochastic Multi-armed Bandits

In the *stochastic* MAB setting the underlying payoff for each action (or *arm*) $a \in \{1, 2, ..., K\}$ is assumed to be independently and identically distributed according to a fixed, but unknown, distribution [13]. That is, at each time-step t the learner chooses some action, a_t, and receives a stochastic payoff drawn from the corresponding distribution, denoted by the random variable $r_{a_t,t} \in \mathbb{R}$. In the query suggestion setting, the actions correspond to suggested queries and the payoffs to some engagement metric with the presented suggestion (i.e. a click). The objective is to choose actions in a way that maximizes the total expected payout or, equiva-

[2]This is roughly equivalent to defining the feature space to be a set of boolean features that uniquely identify each query-suggestion pair.

lently, minimize the *cumulative regret* defined as:

$$R_N \overset{\text{def}}{=} \max_{a=1,..,K} \mathbb{E}\left[\sum_{t=1}^{N} r_{a,t} - \sum_{t=1}^{N} r_{a_t,t}\right] \qquad (1)$$

for a sequence of N actions. Using the assumptions of the stochastic setting, we write this regret as $R_N = N\mu_{a^*} - \sum_{t=1}^{N}\mathbb{E}[\mu_{a_t}] = \sum_{a=1}^{K}\mathbb{E}[T_a(N)](\mu_{a^*} - \mu_a)$, where: μ_a gives the expected payoff of action a; $a^* \overset{\text{def}}{=} \arg\max_{k=1,..,K}\mu_k$ denotes the best action; and $T_a(N)$ gives the total number of plays for action a up until time N.

An important theoretical result in stochastic MAB is the lower bound given by Lai and Robbins [13] who show that the asymptotic regret of any allocation scheme is lower bounded as $\Omega(\log N)$. In the same paper Lai and Robbins present an algorithm, based on the idea of upper confidence bounds (UCB), which (asymptotically) achieves a regret of $\mathcal{O}(\log N)$. This approach was later extended by the UCB1 approach of Auer, et al. [3], which achieves an $\mathcal{O}(\log N)$ regret for any *finite time* and requires only that the payoff distributions have bounded support. Despite the fact that the regret for UCB grows at the best possible rate, improvements in constant factors can make a tremendous difference in practice, which motivates our interest in Thompson sampling.

3.2 Thompson Sampling

Thompson sampling (TS) is a Bayesian MAB approach where the (unknown) payoff parameters are inferred from past data and summarized using a posterior distribution. Actions are then chosen in proportion to their probability of being optimal under this posterior [20]. Specifically, suppose the payoff distribution is given by some parametric likelihood function $P(r|a,\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is a real-valued vector of unknown, unobserved parameters. By treating $\boldsymbol{\theta}$ as a random variable, and assigning some known prior $P(\boldsymbol{\theta})$, we can calculate the posterior distribution over $\boldsymbol{\theta}$ given some observed data $\mathcal{D}$. In the stochastic MAB case, the observed data is given as a sequence of actions and payoffs from time 1 to t, denoted $\mathcal{D}_t \overset{\text{def}}{=} \{(a_1,r_1),...,(a_t,r_t)\}$. Using Bayes rule, as well as the independence of the observed payoffs, the posterior distribution may be written as:

$$P(\boldsymbol{\theta}|\mathcal{D}_t) \propto P(\mathcal{D}_t|\boldsymbol{\theta})P(\boldsymbol{\theta}) = \prod_{i=1}^{t} P(r_i|a_i,\boldsymbol{\theta})P(\boldsymbol{\theta})$$

In order to choose an action at time $t+1$, the Thompson sampling algorithm samples a payoff parameter from the posterior, $\hat{\boldsymbol{\theta}}_{t+1} \sim P(\boldsymbol{\theta}|\mathcal{D}_t)$, then chooses the action that maximizes the expected payoff using this parameter:

$$a_{t+1} = \arg\max_a \int rP(r|a,\hat{\boldsymbol{\theta}}_{t+1})dr$$

Intuitively, this sampling strategy manages the explore-exploit tradeoff almost automatically because the posterior distribution reflects the uncertainty around the parameter vector. That is, if the algorithm is uncertain about some parameters (i.e. the corresponding action payout estimates), their sampled values will range wildly. These actions will potentially be explored, which in turn reduces the posterior uncertainty of the related parameters, permitting more exploitive behavior in the next round.

The drawback of the TS approach is that for many cases sampling from the posterior distribution is not computationally feasible, nor is computing the above integral. However, in the query suggestion task, as well as many web scenarios, the success of a particular recommendation is measured by binary outcomes, e.g., does the user click the link, purchase an item, create a new account. In such cases the payoff distribution for action a may be modelled by a Bernoulli distribution with unknown parameter θ_a (let $\boldsymbol{\theta} = (\theta_1,...,\theta_K)$). By substituting the appropriate likelihood function into the above posterior and assuming a conjugate prior, $P(\theta_a) = \text{Beta}(\theta_a;\alpha,\beta)$ for some practitioner-defined parameters α,β, we arrive at:

$$P(\theta_a|\mathcal{D}_t) = \text{Beta}(S_{a,t}+\alpha, F_{a,t}+\beta)$$

where $S_a \overset{\text{def}}{=} \sum_{t:a_t=a} r_t$ gives the number of "success" events for arm a and $F_a \overset{\text{def}}{=} \sum_{t:a_t=a} 1 - r_t$ the number of "failure" events. This posterior allows us to summarize all past events with only two parameters and can be simulated efficiently. Additionally, it is straightforward to specify prior parameters α and β as they are the number of "successes" and "failures," respectively. In this work, we defer to the de facto standard and use a uniform, relatively *uninformed* prior with $\alpha = 1$ and $\beta = 1$. The pseudocode for this approach is given in Algorithm 1.

Algorithm 1 Thompson sampling for Bernoulli bandits

```
Require: prior parameters (for all arms) $\alpha_a$, $\beta_a$   ▷ default:
    $\alpha = 1, \beta = 1$
Initialize: $S_a = 0, F_a = 0, \forall a$
 1: for each $t = 1, 2, \ldots$ do                          ▷ each round
 2:     for each $a = 1, \ldots, K$ do                      ▷ each action
 3:         Sample $\hat{\theta}_a \sim \text{Beta}(S_a + \alpha_a, F_a + \beta_a)$
 4:     end for
 5:     Observe reward $r_t$ from playing arm $\hat{a} := \arg\max_i \hat{\theta}_i$
 6:     if $r_t = 1$ then
 7:         $S_{\hat{a}} := S_{\hat{a}} + 1$
 8:     else
 9:         $F_{\hat{a}} := F_{\hat{a}} + 1$
10:     end if
11: end for
```

This algorithm, dubbed Bernoulli-bandit Thompson sampling, significantly outperforms other MAB variants empirically [9, 16], but meaningful theoretical analysis had eluded researchers until recently. New work has revealed that, in the Bernoulli-payoff setting, the finite-time regret of TS is $\mathcal{O}(\log(N))$ with constant terms that closely match the Lai-Robbins lower bound [12].

In addition to these strong published results, TS grants a number of problem-specific benefits for the query suggestion task. In particular, new related search suggestions are continually being added to the set of possible choices as new product terms are introduced. In the TS approach, a new action can be added seamlessly: one simply sets initial parameters $S_a = F_a = 0$ and the algorithm begins exploring the new suggestion. In contrast, adding new actions into an UCB implementation, which chooses the action with the highest average plus confidence bound, results in that new action being chosen at every request until the confidence bound shrinks to roughly the same size as the other bounds. In the theoretical analysis of cumulative regret, these strategies are not significantly different, but in terms of user experience the differences are stark.

Lastly, in web-scale systems handling millions of requests daily, it is rarely possible to update the posterior parameters (S_a and F_a) in real time. However, due to its randomized behavior, a TS approach will choose a variety of actions over time even if the posterior parameters are not updated. Again, this contrasts with deterministic UCB approaches, which will continue to choose the same action until the empirical means and confidence bounds are updated.

4. MULTIPLE QUERY SUGGESTIONS

For the case when only a single query suggestion is presented to the user, it is straightforward to apply the Thompson sampling approach (Section 3.2). To begin, it is necessary to construct a manageable number of query-suggestion pairs by eliminating obvious mis-matches. While there are numerous ways to achieve this, the most straightforward way is to choose the top L initial search queries by volume and generate candidates using co-occurrence data. Specifically, we collect the queries entered after the initial query and use the top K, by volume, as the set of suggestions. Here, K should only be kept large enough to ensure we do not miss a suggestion that has a chance of being the optimal choice. Given a manageable set of suggestions, say $\{s_1, ..., s_K\}$, for each query, $\{q_1, ..., q_L\}$, we can deploy L independent instances of the TS approach in Algorithm 1 with suggestions corresponding directly to actions.

However, in most applications there is enough space to supply multiple suggestions to the user at once. For instance, at `ebay.com` up to 10 related search suggestions are displayed at one time; a screenshot of the related search interface is given in Figure 1. Even in the case where we must provide $M > 1$ suggestions per request, it is possible to deploy the same TS algorithm using *meta-actions*. That is, we construct an expanded set of actions where each action corresponds to a unique sequence of M suggestions, which makes M^K actions if duplicates are permitted. This action space is far too large for any practical implementation and so we reduce the size of this action space.

A natural way to reduce the size of the action space is to assume that the probability of user engagement on a suggestion is independent from its position and the other $M-1$ suggestions present. Specifically, if we let $\{z_1, ..., z_M\}$ denote the set of indices, $z_i \in \{1, ..., K\}$, for the suggestions present on some request, and $(\theta_1, ..., \theta_K)$ denote the Bernoulli parameters describing the engagement probability on each suggestion, the likelihood function is as follows:

$$\begin{aligned} &P(r|(s_{z_1}, ..., s_{z_M}), (\theta_1, ..., \theta_K)) \\ &= \sum_{i=1}^{M} P(r_i|(s_{z_1}, ..., s_{z_M}), (\theta_1, ..., \theta_K)) \\ &= \sum_{i=1}^{M} P(r_i|s_{z_i}, \theta_{z_i}) = \sum_{i=1}^{M} \theta_{z_i} \end{aligned} \tag{2}$$

where $r_i \in \{0, 1\}$ is an indicator variable for an engagement at position i and $r = \sum_i r_i$. Using this likelihood we arrive at a TS approach very similar to Algorithm 1 except that the M suggestions with the largest engagement rate, according to a sampled θ, are shown to the user instead. The Beta posterior updating remains unchanged.

Additionally, we inject two modifications to the algorithm based on our observations of scenarios in practice. First, in line 12, we propose to update $F_{a_{z_m}}$ by $\frac{1}{M-1}$ instead of 1.

Algorithm 2 M-Independent arm Thompson sampling

```
Require: prior parameters (for all arms) $\alpha_i$, $\beta_i$   ▷ default: $\alpha = 1, \beta = 1$
Initialize: $S_a = 0, F_a = 0, \forall a$
 1: for each $t = 1, 2, \ldots$ do                              ▷ each round
 2:     for each $a = 1, 2, \ldots, K$ do                      ▷ each arm
 3:         Sample $\hat{\theta}_a$ from the $Beta(S_a + \alpha, F_a + \beta)$
 4:     end for
 5:     $(z_1, ..., z_M) \leftarrow top_M(sorted\ (\theta_i))$     ▷ Select $M$ items
 6:     Observe rewards $(r_1, \ldots, r_M)$ from playing arms $(a_{z_1}, \ldots, a_{z_M})$
 7:     for each $m = 1, 2, \ldots, M$ do                      ▷ each arm
 8:         if $r_m = 1$ then                   ▷ Success on the arm $z_m$
 9:             $S_{a_{z_m}} = S_{a_{z_m}} + 1$
10:         else
11:             if $r = 1$ then ▷ Observing a reward at some arm, $r = \sum_i r_i$
12:                 $F_{a_{z_m}} = F_{a_{z_m}} + \frac{1}{M-1}$
13:             else
14:                 $F_{a_{z_m}} = F_{a_{z_m}} + \frac{\gamma}{M}$
15:             end if
16:         end if
17:     end for
18: end for
```

Namely, we assign "one" failure (but not $M-1$ failures) if we confer one success to some arm at each round, and the failure is shared by all the remaining arms. This helps prevent potential under-exploration that might occur as a result of our independence assumptions: otherwise, we might shift a bell curve (a prior) to the left too quickly. Second, we introduce the rate parameters γ (line 14) to accommodate the ignorance of not observing a user action. Intuitively, γ is a scale factor to control the rate at which the posterior uncertainty decreases when no user decision is spotted. For example at $\gamma = M$, the algorithm sees no-response as dissatisfaction.

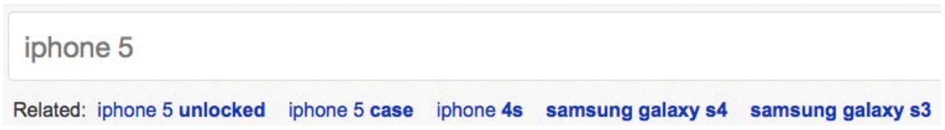
5. EXPERIMENTS

We now consider an online related search application. Given a user issuing a query to an e-commerce site, the goal is to prompt the best related queries (suggestions) for that user. There are various ways of measuring performance, and here we adopt the click-through rate (CTR) estimation, i.e. what is the probability that a user is interacting with the related searches module? Intuitively, one may consider a click as a user's recognition of valuable or useful information.

5.1 Setup

Ideally, the best way of running experiments is to deploy our algorithm in production and observe the outcome. However, without evidence of potential success, it is infeasible to deploy an algorithm to `ebay.com` because millions of transactions are potentially affected.

We evaluated our work by replaying user actions recorded in `eBay` logs. To start, we extracted user activities related to query transitions (changes). Specifically, each transition consists of a triplet (q, s, r) where q refers to some query, s refers to q's immediately successor ($q \neq s$) in the same user session, and r refers to its reward. If s is the result of the related search module displayed in the search result page of


iphone 5

Related: iphone 5 **unlocked** iphone 5 **case** iphone **4s** **samsung galaxy s4** **samsung galaxy s3**

Figure 1: Related Search on `ebay.com`

q, the reward $r = 1$; otherwise, $r = 0$. For example, in Figure 1, if a user clicks on “iphone 4s”, the triplet is (q =“iphone 5”, s =“iphone 4s”, $r = 1$), and if a user manually types a new query “htc”, instead of interacting with related search module, the triplet is (q =“iphone 5”, s =“htc”, $r = 0$). For simplicity, all data are case insensitive.

We conducted experiments based on data collected from the first two weeks of November, 2013. We evaluated our algorithm against the top-100 popular queries based on counting transitions of $q \rightarrow s$ in terms of related-search clicks. The top-100 queries guarantee hundreds of thousands of appearances to eBay users and thousands of clicks on related searches. Thus, the CTR of s given q (which later serves as the ground truth) across two weeks are reliable. To protect business sensitive information, the queries selected are not reported, but one may expect that the top popular queries are often short (2-3 words), e.g. “`iphone 5`,” “`xbox 360`,” “`watch`,” “`dress`,” “`ps4`,”.

In sum, to evaluate our concept, we use the following strategy: suppose that we displayed I candidates. If we were only allowed to display M suggestions, can our algorithm identify the best candidates? That is, the solution should correctly identify M suggestions with high CTRs from I choices.

5.2 Evaluation

The objective of the algorithm is to maximize the total accumulated rewards based on determining which related searches to be displayed. Analogously, it aims to minimize the expected cumulative regret R_N, or the total loss caused by not displaying a set composed of the best candidates:

$$\mathcal{R}_{\mathcal{T}} \overset{\text{def}}{=} \sum_{t=1}^{N} (\mathbb{E}_M^* - \mathbb{E}_M) \tag{3}$$

where $\mathbb{E}_M^*$ indicates the optimized CTR, i.e. the expected CTR when a set of best M choices with highest CTRs are displayed, and $\mathbb{E}_M$ is the measured CTR from display arms $(z_1, ..., z_M)$ (line 5 of Algorithm 2).

To compute $\mathbb{E}_M$, or the CTR of a set of M suggestions being displayed, we assume related searches are independent from each other with respect to CTR, and thus we have $\mathbb{E}_M = \sum_{z \in (z_1,...,z_M)} \mathbb{E}(z)$. The best choice, composed of related searches with the highest $\mathbb{E}(z)$, then naturally refers to those queries with their CTRs ranking in the top in that their sum becomes the largest.

It is worth mentioning that $\mathbb{E}(z)$, the expected CTR of the candidate z to the query q, is often not constant over a long period of time. A breaking event could affect $\mathbb{E}(z)$. For example, it is expected to see a difference on the $\mathbb{E}(z =$ “`iPhone 4`”) for the query “`iPhone`” before the release of iPhone 5, versus after it. To alleviate concerns over fluctuating CTRs, in our experiments we constrain to 2 weeks. We focus on popular queries because we can collect enough user feedback to evaluate our work.

5.3 Discussion

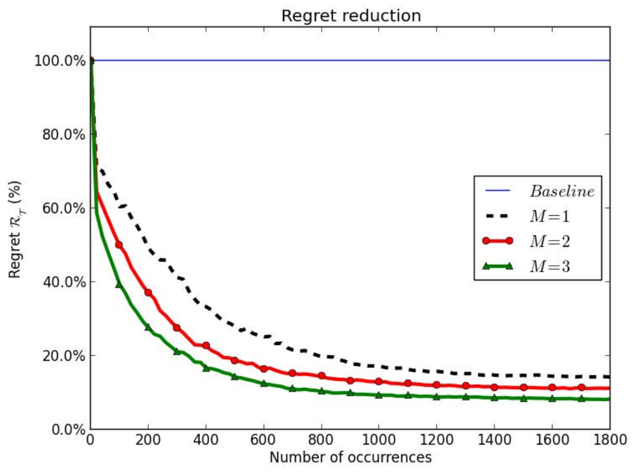


Figure 2: Regret reduction curve ($\gamma = 0.02$)

Figure 2 summarizes the results of three different settings of M. All three settings follow a similar trend (declining in $\mathcal{R}_{\mathcal{T}}$) when applying Algorithm 2. Under the hood, we replay all user decisions in the same order as they were in our search log. The x-axis represents the moment of x_{th} occurrence of displaying related searches; the y-axis shows the average $\mathcal{R}_{\mathcal{T}}$ of all queries we analyzed against a baseline in percentage scale. The baseline refers to the expected regret $\mathcal{R}_{\mathcal{T}}$ (Equation 3) when $\mathbb{E}_{\mathcal{I}}$ is always selected randomly. Intuitively speaking, it is the cost (i.e. CTR loss) paid if we explore all candidates evenly; the decline indicates the selection of candidates with better CTR over time.

It is natural to expect a smaller $\mathcal{R}_{\mathcal{T}}$ when increasing M because the chance of identifying good candidates in three slots ($M = 3$) is higher than in one slot ($M = 1$), especially at the early phase of learning. That is, a worse choice means a greater loss when $M = 1$ than when $M = 3$. We also notice that around 400 $\sim$ 800 on the x-axis the decreasing process reaches a plateau, signifying rapid convergence when applying the algorithm in practice because 400 occurrences can be achieved for most queries on a site like eBay. Moreover, a large drop in the early phase ($<$ 100 on the x-axis) suggests its potential when dealing with rare queries.

An immediate next step is to investigate the penalty factor γ in Algorithm 2. The choice of γ reflects one’s belief toward a user’s rewriting a query instead of interacting with related searches. Setting $\gamma = 0$ suggests complete ignorance to rewritings, i.e., one may consider rewritings as shifts in intent or other unrelated user actions. In contrast, setting $\gamma = M$ treats a rewriting as a failure, i.e. all candidates displayed to users then receive an even penalty.

In Figure 3, we summarize our findings as regards γ by highlighting four choices. Applying penalties ($\gamma > 0$) generally leads to a faster descent in the early stages, and it may

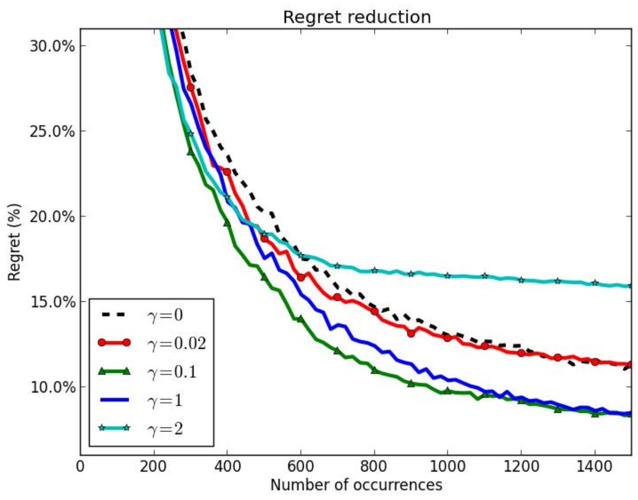


Figure 3: The choice of gamma ($M = 2$)

also lead to less regret eventually (in this example, $\gamma = 0.1$ or 1). However, penalizing too much brings negative effects. For example, $\gamma = 0.1$ reduces regret faster than $\gamma = 1$. If we consider no user feedback as a failure of the classical TS proposed, i.e. $\gamma = M = 2$, it leads to even worse results. Apparently, the optimized γ depends on the type of a reward and the chance of observing of a reward. In our experiments where the reward is a user's clicking and its chance is CTR, we found $\gamma = 0.05 \sim 0.25$ to be a decent range based on heuristic trials. This deserves further study in that with a better understanding of user satisfaction, we may associate γ with other factors, like dwell time or bounce rate.

6. CONCLUSION AND FUTURE WORK

In this paper, we investigated the possibility of viewing related searches in a e-commerce search as an exploration/exploitation problem. By collecting user-feedback as rewards, we extended the idea of Thompson sampling to provide functionality for continuously refining related searches. We also investigated the consideration of user rewriting behaviors, and conducted a heuristic study of how to achieve less regret by tuning a penalization factor. The proposed algorithm requires only one parameter to tune, and thus is easy to implement and to scale. Our historical study also confirms its potential and robustness in practice.

Future work includes a further theoretical analysis of its finite-time regret, testing in a production environment to measure true regret, a deterministic study on the parameter γ, removal of the independence assumption among related searches, and modifications of the algorithm to accommodate rare queries. Furthermore, if introducing new related search candidates is done routinely, one may consider a mechanism to explore newly introduced candidates, or to gradually "forget" obsolete information.

Acknowledgement

We thank Yoni Medoff for his comments and feedback on this paper and Chris Severs and Sunny Li for their insights on improving related searches.

7. REFERENCES

[1] S. Agrawal and N. Goyal. Analysis of thompson sampling for the multi-armed bandit problem. *CoRR*, 2011.

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

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

[4] R. Baeza-Yates, C. Hurtado, and M. Mendoza. Query recommendation using query logs in search engines. In *Current Trends in Database Technology-EDBT 2004 Workshops*, pages 395–397. Springer, 2005.

[5] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 407–416. ACM, 2000.

[6] P. Boldi, F. Bonchi, C. Castillo, D. Donato, and S. Vigna. Query suggestions using query-flow graphs. In *Proceedings of the 2009 workshop on Web Search Click Data*, pages 56–63. ACM, 2009.

[7] S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5, 2012.

[8] S. Bubeck and C.-Y. Liu. Prior-free and prior-dependent regret bounds for thompson sampling. In *Advances in Neural Information Processing Sys.*, pages 638–646, 2013.

[9] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, editors, *25th Annual Conference on Neural Information Processing Systems*, NIPS '11, pages 2249–2257, 2011.

[10] C.-K. Huang, L.-F. Chien, and Y.-J. Oyang. Relevant term suggestion in interactive web search based on contextual information in query session logs. *Journal of the American Society for Information Science and Technology*, 54(7):638–649, 2003.

[11] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 387–396, New York, NY, USA, 2006. ACM.

[12] E. Kaufmann, N. Korda, and R. Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In *Proceedings of the 23rd International Conference on Algorithmic Learning Theory*, ALT'12, pages 199–213, Berlin, Heidelberg, 2012. Springer-Verlag.

[13] T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[14] L. Li, W. Chu, J. Langford, and R. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010.

[15] T. Lu, D. Pál, and M. Pál. Contextual multi-armed bandits. In *International Conference on Artificial*

Intelligence and Statistics, pages 485–492, 2010.

[16] B. May and D. Leslie. Simulation studies in optimistic bayesian sampling in contextual-bandit problems. Technical report, Technical Report 11: 02, Statistics Group, Department of Mathematics, University of Bristol, 2011.

[17] U. Ozertem, O. Chapelle, P. Donmez, and E. Velipasaoglu. Learning to suggest: a machine learning framework for ranking query suggestions. In *Proceedings of the 35th international SIGIR conference on Research and development in information retrieval*, pages 25–34, 2012.

[18] F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th international conference on Machine learning*, pages 784–791. ACM, 2008.

[19] S. L. Scott. A Modern Bayesian Look at the Multi-armed Bandit. *Appl. Stoch. Model. Bus. Ind.*, 26(6):639–658, 2010.

[20] W. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, pages 285–294, 1933.